Intelligence artificielle :
Les agents intelligents

Travail réalisé dans le cadre du cours d'intelligence artificielle de Mr. Giot

Année Académique 2004-2005
Etudiants en 4ème Informatique

Olivier Bibot
Vincent Brakeva

Table des matières

Introduction

Dans les années 70, en pleine vague d'Intelligence (très) Artificielle, on espérait que les Agents soient des unités logicielles douées de facultés d'adaptation et de collaboration. D'où la création du terme : "Agent Intelligent".

Mais qu'est ce qu'un agent ? Qu'appelle t'on intelligence ? Où en sommes nous à l'heure actuelle ? Toutes ces questions nous y répondrons, et à bien d'autres encore, pour ce faire nous avons découpé notre dossier en différents chapitres...

Dans le premier, nous allons poser la définition de quelques termes importants concernant les agents ainsi que leurs caractéristiques et leur classification. Le second vous expliquera les principes de base, les principales structures et nous en développerons certaines pour montrer leur architecture et leur mécanisme. Le troisième chapitre abordera les systèmes multi-agents et plus particulièrement la communication, la négociation et les interactions entre les différents agents. La quatrième partie sera consacrée à la mise en œuvre des agents (les langages utilisés). Nous évoquerons ensuite les problèmes de sécurité, d'éthique et les perspectives d'avenir.

On vous souhaite une bonne lecture en espérant que vous prendrez autant de plaisir que nous en avons pris pendant la rédaction de ce document.

1 - Généralités

1. Définition des termes et notions :

Qu'est-ce qu'un agent ?

Si on se réfère à la définition du dictionnaire : du latin "agens" : celui qui agit.
"*Un agent est une personne chargée des affaires et des intérêts d'un individu, d'un groupe ou d'un pays, pour le compte desquels elle agit*".

Le terme "agir" est défini par le petit Larousse : faire quelque chose, s'occuper, produire un effet.

De ces définitions on peut retirer deux aspects fondamentaux :

- un agent accomplit quelque chose,
- un agent agit à la demande de quelqu'un (Agent ou utilisateur).

Comme dans le cas de toute technologie nouvelle il n'y a pas de définition universelle, mais de multiples. Citons, par exemple, la définition donnée par Caglayan et Harrison (Définition donnée dans le livre "Les Agents" de Alper Caglayan et Colin Harrisson aux Editions InterEditions, 1997) :
Agent logiciel : entité informatique qui réalise de manière autonome des tâches pour un utilisateur.

On peut conclure de ces définitions qu'un agent informatique (plus exactement une application agent) devra faire quelque chose pour une personne ou une application.

Intelligent

"L'intelligence est l'aptitude à comprendre, à donner un sens et à s'adapter à une situation, à choisir en fonction des circonstances" (définition du petit Larousse).

Pour un agent, cette définition pose énormément de problèmes car il est très difficile de caractériser cette "aptitude à comprendre et à s'adapter à une situation nouvelle". De nombreux auteurs ont donné leurs visions des choses mais aucune de ces visions n'est reconnue (un auteur = une définition).
On décrira donc l'intelligence d'un agent comme un ensemble de caractéristiques : la capacité d'apprendre, la capacité sociale et un haut degré d'autonomie.

Le débat est interminable, peut-être insoluble, et sûrement sans grand intérêt pour l'utilisateur final. Beaucoup de sociétés, pour d'évidentes raisons de marketing, profitent de ce flou terminologique pour qualifier leurs produits logiciels d'Agents intelligents. Ce qui est clair, c'est qu'à l'heure actuelle, aucun agent dit "intelligent", ne possède l'ensemble des caractéristiques de l'intelligence.

Agent intelligent

Un agent intelligent sera donc une application orientée tâche, c'est à dire qu'elle déploiera une activité (suite de fonctionnalités offertes par son environnement) dans le but de faire quelque chose et sera caractérisée par un certain degré d'autonomie, d'interactivité et de réactivité.

2. Les caractéristiques des agents :

2.1. Caractéristiques fondamentales

L'autonomie : L'agent travaille sans intervention directe, jusqu'à un point défini par l'utilisateur. L'autonomie d'un agent peut aller du simple lancement d'une sauvegarde la nuit, à la négociation du prix d'un produit choisi par son mandataire.

L'interactivité : L'agent doit pouvoir exercer des actions sur son environnement et réciproquement. L'interactivité d'un agent de sauvegarde sera sa possibilité de sauvegarder un certain nombre de fichiers par une action exercée au travers du système d'exploitation. Ce dernier pourra informer l'agent d'une permission non accordée (exemple : un fichier dont l'utilisateur n'a pas de droit de lecture).

La réactivité : L'agent percevra son environnement (qui pourra être l'utilisateur au travers une interface graphique, un ensemble d'agents, …) en répondant dans les temps impartis aux changements qui surviennent sur cet environnement (exemple : un agent de sauvegarde pourra faire sa tache à une heure donnée).

2.2. Caractéristiques d'intelligence

La capacité d'apprendre

Le petit Larousse définit "apprendre" (du latin appréhendée, saisir) comme le fait "d'acquérir la connaissance, l'information, l'habitude". Un agent aura la capacité d'apprendre si il sait acquérir de la connaissance, de l'information ou des habitudes.

Exemple : Un agent grâce à sa capacité de réactivité, doit se déclencher à une certaine heure. Mais l'utilisateur l'arrête dans sa tache (qui pourrait être une sauvegarde) car il ralentit le travail de l'utilisateur. L'agent va apprendre à différer son exécution pour éviter de gêner l'utilisateur.

La capacité sociale

Les agents interagissent avec les autre agents (et éventuellement des êtres humains) grâce à des langages de communication entre agents. Cette capacité sera la base pour la coopération entre les agents.

Exemple 1 : Notre agent de sauvegarde rencontre un autre agent de sauvegarde sur un réseau. Ces deux agents peuvent se mettre d'accord pour se partager le travail afin que la tâche soit achevée plus vite.

Exemple 2 : L'agent ne peut exécuter sa tâche pour des raisons techniques diverses (problèmes de droits d'accès, par exemple). L'agent va communiquer se fait à l'utilisateur, en lui apportant la démarche à suivre dans un langage naturel compréhensible. L'utilisateur pourra ensuite, en utilisant le même langage, indiquer à l'agent comment résoudre le problème.

Haut degré d'autonomie

L'agent fonctionne sans intervention directe humaine ou autre (autonomie) et en plus il a une forme de contrôle sur ses actions et sur leur état interne (haut degré).

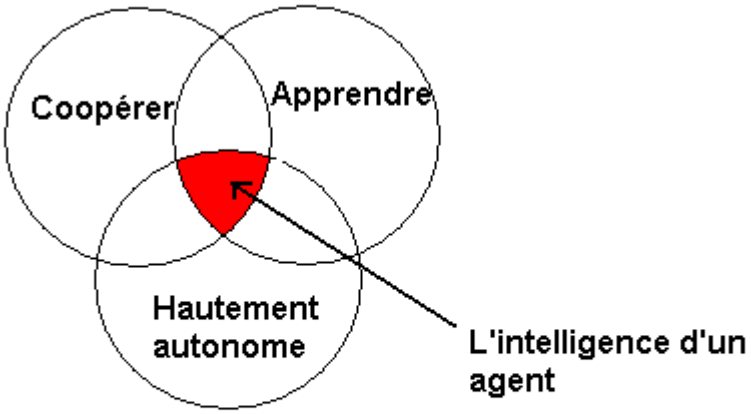

Figure 1 - Schéma de la représentation de l'intelligence d'un agent

Un peu plus d'intelligence

La pro-activité est une caractéristique qui est intégrée dans les agents hautement intelligents (très rares actuellement). Les agents n'agissent pas seulement en réponse à leur environnement, mais ils sont capables d'avoir un comportement guidé par un but, en ayant la possibilité de prendre l'initiative.

Exemple : Un agent réseau peut décider, de lui-même, pendant un temps où il est non actif, de faire des statistiques sur les routeurs pour améliorer son activité future.

Cette notion de pro-activité peut être décrite comme l'ensemble des qualités suivantes : croyances, désirs, buts, intentions, choix, décisions, engagements, conventions et obligations

Exemple : Explication de la pro-activité avec la "vie d'un humain", ici Pierre

- connaissances — Pierre connaît le fait que les humains sont mortels
- croyances — Pierre a pris son parapluie parce qu'il croit qu'il va pleuvoir
- désirs, buts — Pierre désire avoir son doctorat
- intentions — Pierre a l'intention de travailler dur pour avoir sa thèse
- choix, décisions — Pierre a décidé de faire une thèse
- engagements — Pierre ne va pas s'arrêter de travailler avant d'avoir fini sa thèse
- conventions — si, par hasard, Pierre décide d'abandonner sa thèse, il va le dire à son professeur
- obligations — Pierre doit travailler pour entretenir sa famille

2.3. D'autres caractéristiques

Les caractéristiques suivantes peuvent être ajoutées aux caractéristiques de base pour permettre à l'agent d'exécuter sa tâche. Il s'agit ici d'une liste non exhaustive, d'autres caractéristiques peuvent s'ajouter et se combiner.

Coordinatif

L'agent est capable de coordonner ses actions par rapport à un utilisateur ou un autre agent.

Exemple : Un agent de sauvegarde sauve les données qui viennent d'être mises à jour par un utilisateur ou par un autre agent.

Compétitif

L'agent est capable d'agir dans un environnement où d'autres agents interviennent. Le but est le même pour tous les agents présents, mais un seul l'atteindra. Les autres échoueront et, forcément, tous les coups sont permis.

Exemple : Un agent commercial peut chercher les meilleurs prix et les meilleurs services pour un produit donné. En négociant avec les fournisseurs, d'une manière plus rapide et plus optimale qu'un autre agent ou utilisateur. Un agent très compétitif pourra induire ses "adversaires" sur de fausses pistes.

2.4. Mobilité

Nous devons prendre en compte deux types de mobilités :

- ***Mobilité relative***, ou par requêtes : Dans ce cas il n'y a pas un réel déplacement de l'agent. Celui-ci lance une succession de requêtes à destination de différents serveurs. C'est le cas des agents de recherche, tel que Copernic, qui interroge différents moteurs de recherche afin de fournir à son utilisateur une synthèse des résultats.
- ***Mobilité réelle*** de l'agent : Le processus agent se déplace d'un serveur à un autre, sur le réseau. Le code de l'objet est transporté et ses données, aussi. Ensuite, il continue son exécution sur la nouvelle machine.

Exemple : Un agent de sauvegarde peut se déplacer sur plusieurs serveurs dans le but de faire des restaurations de fichiers.

Quand nous parlerons d'agent mobile, c'est ce dernier cas qui sera pris en compte.

3. Les types d'agents

3.1. Les agents réactifs et cognitifs

On distingue deux catégories d'agents ayant un mode de fonctionnement bien différent :

Les **agents réactifs** sont souvent qualifiés de ne pas être " intelligents " par eux-mêmes. Ils sont des composantes très simples qui perçoivent l'environnement et sont capables d'agir sur celui-ci. Ils n'ont pas une représentation symbolique de l'environnement ou des connaissances et ils ne possèdent pas de croyances, pas de mécanismes d'envoi de messages.
Leurs capacités répondent uniquement au mode stimulus/action qui peut être considéré comme une forme de communication.

Les **agents cognitifs** sont plus évolués. Ils sont le résultat direct des recherches menées dans le domaine de l'intelligence artificielle. Ils sont donc principalement caractérisés par un niveau symbolique de la représentation des connaissances et par des notions mentales.
Les agents cognitifs possèdent une représentation partielle de l'environnement, des buts explicites, ils sont capables de planifier leur comportement, mémoriser leurs actions passées, communiquer par envoi de messages, négocier, etc.

3.2. Classification des agents suivant leur fonctionnalité

La technologie agent est bouillonnante. De très nombreux acteurs interviennent sur le sujet : universités, centres de recherche, sociétés privées et leur classification des agents dépend de leur utilisation. L'importance de cette classification est qu'elle permet de donner des attributs aux agents.
On retiendra celle du laboratoire de British Telecom (1996) qui distingue 4 types d'agents : les agents d'interface, les agents collaboratifs, les agents collaboratifs capables d'apprentissage et les agents réellement intelligents.
La société Reticular System Inc qui produit le Toolkit AgentBuilder (www.agentbuilder.com) a aussi sa classification. Elle s'appuie sur la précédente mais y ajoute des agents mobiles, des agents d'information et des systèmes d'agents hétérogènes.

Mais depuis , de nouveaux types d'agents sont apparus et ces classifications ne suffisent plus. Aussi, nous allons prendre cette nouvelle classification :

- Les agents intelligents,
- Les agents mobiles,
- Les agents d'information,
- Les agents d'interface utilisateur,
- Les agents commissionnaires,
- Les systèmes multi-agents.

Et, pour compliquer un peu plus, chaque type d'agent peut se diviser en plusieurs sous-types.

Il ne faut pas oublier que cette classification est susceptible d'évoluer chaque jour grâce aux nouvelles percées dans le domaine des agents intelligents

3.3. Les agents intelligents

On peut dire que la notion d'intelligence pour les agents est complètement différente de celle des humains. Les agents ne sont pas là pour supplanter les humains, mais pour les assister. Un agent intelligent doit avoir de la connaissance et doit pouvoir agir dessus. Il peut examiner ses objectifs, nommer ses intentions, planifier ses actions et éventuellement, agir sur ses plans. De plus , il doit être capable d'interagir avec d'autres agents.
Cela ressemble beaucoup à un comportement humain. Il ne faut pas être surpris, les chercheurs utilisent leur connaissance de l'intelligence humaine comme modèle de l'intelligence des agents.

3.4. Les agents mobiles

Les agents mobiles se déplacent d'une machine à l'autre. Pour migrer, un agent mobile transfère son code et ses données sur le nouveau site puis continue son exécution sur ce site là. A l'issue de la migration, le processus de la machine initiale est détruit par une commande du nouveau processus.
L'intérêt de ces agents concerne les recherches dans beaucoup de données sur des sites éloignés. Par contre, les agents mobiles entraînent des problèmes de sécurité pour le site hébergeur, mais aussi pour l'agent. Ils entraînent des problèmes de consommation de ressources et de dimensionnement sur la machine hôte.

3.5. Les agents d'information

« Interest Matching Agents »

Les « interest matching Agents » sont probablement les agents les plus utilisés et la plupart des utilisateurs ne savent pas qu'ils les utilisent. On retrouve ces agents sur les sites WEB commerciaux. Ils y proposent des recommandations (recommandations provenant de www.fnac.fr) : "Vous avez aimé le livre "Intelligence artificielle et informatique théorique" de Jean-Marc Alliot et Thomas Schiex, vous aimerez certainement le livre "Les Réseaux neuromimétiques" de Jean-François Jodouin".
Ces agents, basés sur les travaux de Patti Maes au MIT (Massachusetts Institute of Technology) Media Laboratory, observent des comportements similaires et des habitudes pour faire leurs recommandations.

Les agents de recherche d'information

La quantité d'information disponible restreint la capacité de la plupart des utilisateurs de retrouver l'information utile. Les agents de recherche disposent de la connaissance de diverses sources d'information. Cette connaissance inclut le type des informations disponibles à chaque source, comment accéder à ces données et leur fiabilité.

Les agents de filtrage d'information

Une autre tâche commune des agents. Les agents de filtrage d'information essaient de résoudre le problème de la surcharge d'information en limitant et en triant les informations arrivant à un utilisateur. L'idée de base est de développer un substitut en-ligne qui connaît suffisamment les goûts de son utilisateur pour choisir les documents intéressants. Ces agents

sont parfois incorporés à des agents de recherche pour éviter des résultats de recherche trop importants. Les agents de filtrage d'information comportent des mécanismes d'apprentissage. Cela leur permet de s'adapter aux besoins de leur utilisateur.

Les agents de suivi de l'information

Des tâches sont dépendantes de la notification du changement de certaines données. La veille technologique utilise ce genre d'outils pour suivre les changements sur les sites WEB choisis. Un planificateur de logistique du transport aérien peut être informé des changements dans les conditions météorologiques et dérouter ses avions en conséquence.
Les agents de suivi de l'information permettent de consulter des sources de données diverses et de faire un suivi permanent des changements apportés. Ces agents pourraient être mobiles pour pouvoir se déplacer de site en site ou pour se rendre dans des endroits plus difficilement accessible.

Les agents de médiation de sources de données

Le paysage de la gestion des données est rempli d'une multitude de systèmes. La plupart ne se parlent pas. Les agents peuvent être employés comme des médiateurs entre toutes ses sources de données, fournissant les mécanismes leur permettant d'inter-opérer. Ces mécanismes peuvent être un protocole de communication et des ontologies décrivant les données contenues dans ces sources.

3.6. Les agents d'interface utilisateur et les assistants personnels

Un agent d'interface est un programme qui assiste l'utilisateur dans la mise en œuvre d'un système ou d'un logiciel (Voir les agents office de Microsoft). Dans le futur, ces agents pourront être étendus au monde du WEB, pour assister l'utilisateur.

3.7. Les agents d'analyse de l'offre

Tous ceux qui ont fait l'expérience de rechercher un article sur le WEB (exemple un CD de musique) dans le but de comparer les offres ont mesuré la difficulté et la longueur de l'opération. Il est nécessaire de trouver des sites WEB spécialisés dans la vente de disques, de déterminer si le titre recherché y est référencé, d'en trouver le prix et ceci pour chaque site WEB. Toute personne ayant tenté une telle opération finit par le regretter...

Or, c'est typiquement le genre de problèmes qu'un agent intelligent peut prendre en charge à la place d'un utilisateur.

Ces agents pourront donc vous renseigner :

- sur la disponibilité d'un produit en menant une recherche par marque ou par catégorie (produits + accessoires),
- sur l'identification des distributeurs : localisation d'un distributeur précis, liste intégrale ou sélective de distributeurs (en fonction des services qu'ils offrent : garantie, facilité de paiement,...),
- en traitant les informations collectées, par exemple grâce à des tableaux comparatifs des offres présentées sur divers critères (prix, services, avis d'autres consommateurs,...).

- en établissant une pré-sélection automatique d'articles en fonction des préférences du consommateur (priorité au rapport qualité - prix, au service, aux avis des autres consommateurs,...)
- en réalisant la transaction :
  - de façon automatique (achat répétitif d'un panier de produits/alimentation, achat dès qu'un modèle est en solde),
  - ou semi-automatique : recommandation/suggestion, accord de transactions automatiques (ordre d'achat, paiement, réception de la facture et gestion simplifiée de la comptabilité)

Bénéfices pour le consommateur : gain de temps, analyse d'une offre commerciale plus étendue, transparence des marchés. Soutien d'un Agent qui connaît de mieux en mieux ses goûts.

Bénéfices pour le distributeur : localisation plus facile des boutiques, augmentation des ventes pour celles qui parviennent à analyser la demande.

3.8. Les systèmes multi-agents

Les agents solos ont très peu d'interaction avec d'autres agents et sont restreints par leurs propres limites. Les systèmes multi-agents (SMA) peuvent profiter des divers rôles de chacun des agents du système.

Un système multi-agent se distingue d'une collection d'agents indépendants par le fait que les agents interagissent en vue de réaliser conjointement une tâche ou d'atteindre conjointement un but particulier.

Un agent qui fait tout serait très difficile à créer, à debugger, à maintenir et il aurait des temps de réponses assez faibles. Découper les fonctionnalités parmi plusieurs sortes d'agents offre une meilleure modularité, flexibilité et extensibilité. Les agents d'un système multi-agents peuvent se "partager" des tâches qui seront exécutées en parallèle. Les agents solos sont plus faciles à construire que les SMA, car les développeurs n'ont pas à se préoccuper de coopération et de coordination. Mais l'industrie aura besoin de ces qualités et donc des systèmes multi-agents.

2 –Structures et architectures des agents

1. Structures des agents intelligents :

1.1. La rationalité des agents :

La vision la plus simple qu'on peut utiliser pour décrire un agent est celle d'une entité qui perçoit son environnement par des détecteurs, et agit sur l'environnement à travers des effecteurs. On aimerait qu'un agent intelligent exécute des tâches pour nous et que, en même temps, il ait un comportement rationnel. Un **agent rationnel** est un agent qui agit d'une manière lui permettant d'obtenir le plus de succès possible dans la réalisation des tâches qu'on lui a assignées.

Comment peut-on mesurer le succès d'un agent "rationnel"?

Pour ce but, il faut disposer d'une **mesure de performance**, si possible objective, associée à une certaine tâche que l'agent doit exécuter. Par exemple, si on construit un agent pour trier notre courrier électronique en quatre catégories : urgent, important, normal et à effacer, on peut mesurer sa performance par le nombre des messages qu'il a classifiés correctement contre le nombre des messages qu'il n'a pas classifiés comme il fallait. On doit aussi décider du moment où il faut mesurer sa performance : faut-il le faire après la classification des messages reçus au cours de l'heure, du jour ou de la semaine écoulée ?

La mesure de performance doit être rapportée aux capacités de l'agent, notamment à ce qu'il a perçu sur l'environnement pendant le temps à considérer (sa *séquence de perception*), à ce qu'il connaît sur son environnement, et aux actions qu'il peut effectuer. La rationalité de l'agent qu'on a construit peut alors être évaluée en tenant compte de tous ces aspects. Pour notre agent trieur de courrier électronique, la perception de l'environnement est l'arrivée des messages, ses connaissances sont les critères de classification, et ses actions sont la répartition des messages dans les quatre catégories et la présentation des messages classifiés à l'utilisateur.

Nous allons voir dans ce qui suit comment on peut conceptualiser et classifier la structure d'un agent par rapport aux attributs qui définissent sa rationalité, et aussi comment on peut évaluer sa performance.

1.2. Structure conceptuelle des agents

Un agent est situé dans un **environnement**. Pour modéliser la structure de l'agent, il faut avoir un modèle de l'environnement. L'environnement peut être vu comme étant dans un état ***e*** parmi un ensemble d'états $E=\{e_1, .., e, ..\}$ L'environnement peut changer son état soit d'une manière spontanée soit comme résultat des actions de l'agent.

L'évolution de l'environnement se modélise différemment selon les caractéristiques que l'on prend en compte, et les simplifications que l'on s'autorise. Les principales distinctions à faire sur les types d'environnements sont :

- **Environnement accessible ou inaccessible**. Un environnement est accessible à l'agent si l'agent peut percevoir entièrement l'état de l'environnement ou, au moins, tous les traits de l'environnement qui sont significatifs du point de vue des actions de l'agent. Sinon, l'environnement est inaccessible.

- **Environnement déterministe ou non déterministe**. Si l'état suivant de l'environnement est déterminé d'une manière unique par l'état courant et l'action de l'agent, alors l'environnement est déterministe. Si le résultat est incertain, notamment si, par suite d'une action de l'agent, l'environnement peut évoluer de différentes manières, alors on est dans le cas non déterministe.
- **Environnement statique ou dynamique**. Si l'environnement ne peut pas changer d'état sans l'intervention de l'agent, on est dans le cas statique. L'environnement est dynamique si son état peut se modifier sans l'action de l'agent dans l'intervalle de temps entre deux perceptions de l'agent.
- **Environnement discret ou continu**. Si tout passage d'un état de l'environnement à un autre nécessite le passage par une séquence d'états intermédiaires, alors on a un environnement continu ; sinon, l'environnement est discret.

Les caractéristiques de l'environnement influencent la façon dont on conçoit un agent car il faut tenir compte de l'évolution de l'environnement, de la capacité de l'agent de saisir cette évolution et de sa capacité à décider en conséquence. Par exemple, si on a plusieurs agents qui agissent dans un même environnement, chaque agent va percevoir l'environnement comme dynamique et non déterministe, car l'état de l'environnement changera en raison des actions des autres agents, et une même action exécutée dans un certain état aura des résultats différents en fonction des actions de ces autres agents.

Nous revenons maintenant à la modélisation des agents : nous commençons par définir une structure simple, puis nous développerons cette structure vers des structures agents plus élaborées.

Au niveau abstrait, on peut identifier quatre types d'agents :
- agents réactifs
- agents avec états
- agents avec buts
- agents avec utilité

1.3. Agents réactifs :

Dans le premier chapitre, on a décrit d'une manière informelle les agents réactifs comme des entités simples basées sur des règles condition-action qui associent des actions aux perceptions de l'environnement.

Maintenant, pour modéliser nos agents, on considère plusieurs fonctions :
- ***voir* : E -> P** est la fonction qui décrit la capacité d'observation de l'environnement, où E est l'ensemble d'états de l'environnement et P est l'ensemble des perceptions de l'agent ;
- ***action* : P -> A** est la fonction qui représente le processus de décision de l'agent : elle détermine quelle action $a \in A$ (A étant l'ensemble des actions disponibles pour l'agent) l'agent choisit en fonction de la perception $p \in P$ qu'il a sur son environnement ;
- ***env* : E x A ->P(E)** est la fonction qui décrit l'évolution de l'environnement ; pour un état $e \in E$ et une action $a \in A$ de l'agent, l'environnement peut modifier son état dans un des états d'un sous-ensemble $E_{e,a}$ ÍP(E), ou P(E) est l'ensemble des sous-ensembles de E.

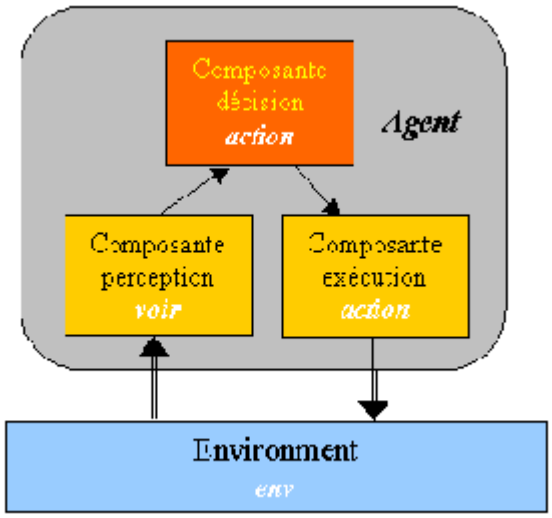

Figure 2 - Structure d'un agent réactif

La structure d'un agent réactif, avec les composantes principales et les fonctions ainsi définies, est représentée dans la figure 2. Il faut noter que, au niveau très général, cette structure correspond aussi aux autres types des agents intelligents, que nous allons définir dans ce qui suit, sauf que chaque composante sera plus développée, ayant sa propre structure.

Si on considère plusieurs agents dans l'environnement, chacun d'eux étant doté de la structure définie ci-dessus, la situation est celle représentée figure 2. Dans ce cas, l'évolution de l'environnement se fait toujours par la fonction ***env***, sauf qu'il y a plusieurs agents qui interagissent avec l'environnement :

- ***env* : E x A_1 x … A_n ->P(E)**, A_j, j =1..n étant l'ensemble des actions disponibles à l'agent j.

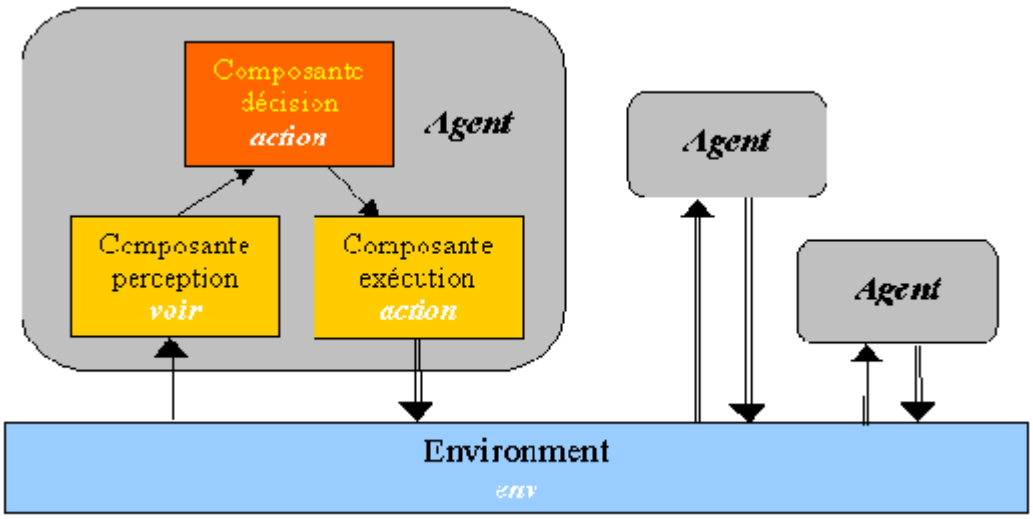

Figure 3 - Structure générale d'un agent dans un système multi-agents

On voit dans la figure 3 qu'un agent interagit uniquement avec l'environnement. En général, les agents réactifs se comportent de cette manière. Mais les agents intelligents peuvent aussi interagir avec les autres agents du système, cette interaction définissant la dimension sociale des agents. Mêmes les agents réactifs peuvent parfois interagir, d'une manière directe ou indirecte, avec les autres agents du système. L'interaction des agents peut être définie comme une spécialisation des fonctions ***action*** et ***voir*** définies précédemment.
On ajoute alors deux autres fonctions à notre modèle :

- ***inter*** **: P -> I** est la fonction qui représente la décision de l'agent concernant son interaction avec un autre agent, où I est l'ensemble des interactions disponibles pour l'agent et $i \in I$ est l'interaction que l'agent choisit (par exemple un message) en fonction d'une perception $p \in P$ qu'il a de l'environnement ;
- ***rinter*** **: I -> A** est la fonction qui décrit la réaction de l'agent, notamment l'action décidée par l'agent, à une interaction transmise par un autre agent.

La nouvelle structure, ainsi augmentée, est présentée dans la figure 3

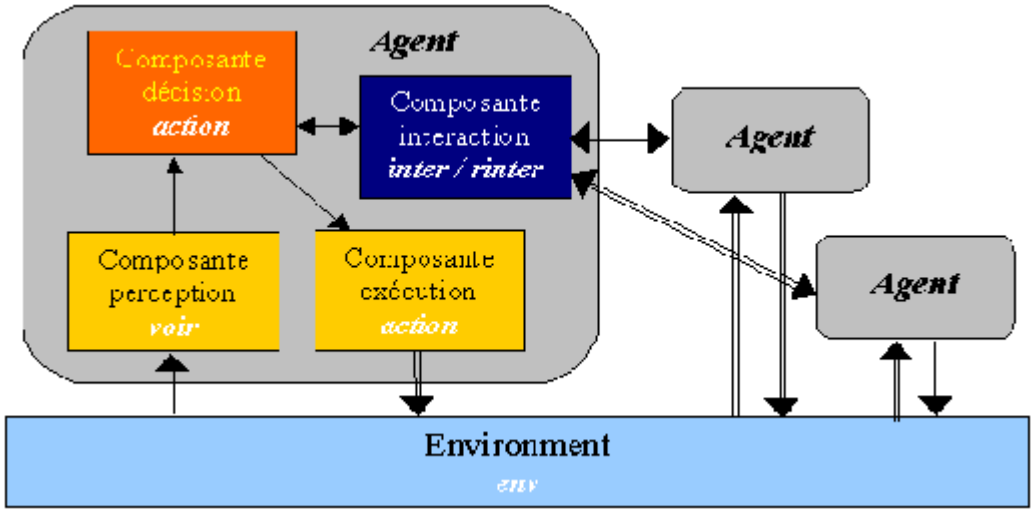

Figure 4 - Structure générale d'un agent qui interagit avec d'autres agents

Il faut noter que la structure présentée dans la figure 4 caractérise les agents intelligents en général, et que la composante interaction avec d'autres agents n'est pas souvent présente dans le cas des agents réactifs. Ce qu'il faut retenir est que cette structure définit aussi les autres types d'agents que nous allons discuter dans ce qui suit.

1.4. Agents avec états :

Les agents avec états, comme leur nom l'indique, maintiennent un état interne qui mémorise la séquence des perceptions de l'agent et, dans certains cas, les actions effectuées par l'agent. L'agent a besoin de maintenir cet état interne pour faire la distinction entre des états de l'environnement qui lui semblent identiques selon la perception qu'il en a, mais qui sont néanmoins différents.

Soit $S=\{s_1, s_2, \ldots\}$ l'ensemble des états internes de l'agent. Dans ce cas, les fonctions de l'agent se modifient de la manière suivante :

- ***action* : S -> A** est maintenant définie sur l'ensemble des états internes de l'agent, ce qui revient à dire que l'agent décide de l'action à effectuer en fonction de son état interne ;
- ***suiv* : S x P -> S** est une nouvelle fonction qui modélise le changement de l'état interne de l'agent en fonction de ses perceptions et de l'état interne courant ;
- ***inter* : S x P -> I** et ***rinter* : S x I -> A** sont modifiées pour montrer que la décision sur les interactions avec d'autres agents dépend aussi de l'état interne ;
- ***voir* : E -> P** et ***env* : E x A ->P(E)** ne sont pas modifiées.

1.5. Agents avec buts

Connaître l'état de l'environnement et la séquence des perceptions mémorisées dans l'état interne n'est pas toujours suffisant pour décider quelle est la meilleure action à effectuer à un moment donné. Au début de ce chapitre, nous avons dit qu'il fallait associer une **mesure de performance** à l'agent. Pour fonctionner d'une manière efficace, l'agent doit essayer d'obtenir le maximum de performance et il doit donc choisir ses actions en conséquence.

Il y a plusieurs façons de définir la mesure de performance. Une première solution est d'indiquer à l'agent ce qu'il doit faire en toute circonstance pour avoir du succès. C'est le cas des agents réactifs, où le concepteur définit les règles condition-action pour chaque cas de perception sur l'environnement. Toutefois, puisque les agents intelligents sont autonomes et proactifs, on aimerait n'avoir à dire à notre agent que ce qu'il faut faire, sans avoir à lui dire exactement comment le faire dans chaque situation. Dans ce cas, une deuxième solution pour définir la mesure de performance de l'agent est de fixer des **états-but** ou états désirables :

- ***but* : E -> {0, 1}** est une fonction qui a la valeur 1 pour les états-but et 0 pour les autres.

Si les états-but sont connus d'avance, on peut définir l'ensemble **B** des buts de l'agent, **B** Ì **E**. Par exemple, un robot qui ramasse des objets sur une surface pour les transporter à la base sait que l'état-but est l'état "parvenir à la base avec le maximum d'objets". Si les états-but ne sont pas connus mais que l'on connaît uniquement certaines propriétés désirables de ces états, alors la fonction ***but*** fait l'évaluation de chaque état pour voir si l'état possède ou non ces propriétés. Par exemple, si le robot ne connaît pas d'avance la position de la base mais qu'il peut la percevoir une fois arrivé près d'elle, le robot doit tester dans chaque position s'il est près de la base ou pas.

Une troisième solution pour définir la mesure de performance de l'agent est décrite dans ce qui suit.

1.6. Agents avec utilité

La mesure de performance d'un agent peut être définie d'une manière plus fine en associant à chaque état de l'ensemble **E** une valeur réelle, l'utilité, qui mesure la désirabilité de cet état pour l'agent. On peut voir les états-but comme un cas particulier des états avec utilité, puisque cela correspond au cas où l'utilité ne peut prendre que les valeurs 1 et 0. Pour associer une utilité aux états, on introduit une nouvelle fonction :

- ***utilité* : E -> R**, où R est l'ensemble des nombres réels.

La fonction ***utilité*** est mieux adaptée que la fonction ***but*** dans deux situations. Premièrement, si l'agent a des buts contradictoires, par exemple l'agent Pierre veut aller au cinéma et étudier pour son prochain examen, la fonction ***utilité*** indique le but à choisir. Deuxièmement, si l'agent agit dans un environnement non déterministe, il n'est pas toujours certain de pouvoir atteindre ses buts. Dans ce cas, l'utilité des buts offre un moyen de rapporter la probabilité de succès à l'importance des buts. En général, dans un environnement non déterministe, l'agent va souvent avoir besoin de la fonction ***utilité*** pour prendre des décisions sur les diverses actions à effectuer tout en sachant que les résultats de ces actions ne sont pas certains.

Dans un environnement non déterministe, ***env* : E x A ->P(E)**, le résultat de l'action **a**, notamment l'état suivant est un élément du sous-ensemble *env*(e,a) qui contient les états suivants possibles après l'exécution de l'action **a** dans l'état **e** de l'environnement. L'utilité d'un état peut être combinée avec le résultat probable d'une action pour déterminer l'utilité attendue d'une action. Avant l'exécution d'une action **a**, l'agent associe une probabilité à chaque résultat possible de cette action, et nous noterons prob(*ex*(a,e) = e') la probabilité estimée par l'agent que le résultat de l'exécution de l'action **a** dans l'état **e** sera l'état **e'**. On a bien évidemment :

$$\sum_{e \in env(e,a)} prob\left[ex(a,e)=e'\right]=1$$

L'utilité attendue d'une action dans l'état **e** du point de vue de l'agent est alors calculée avec la formule :

$$UA(a,e)=\sum_{e \in env(e,a)} prob\left[ex(a,e)=e'\right]*utilité(e')$$

Le principe de la plus grande utilité attendue - utilité attendue maximale, en anglais Maximum Expected Utility (MEU) - dit qu'un agent rationnel devrait choisir l'action qui lui apporte la plus grande utilité attendue. En pratique, le calcul des probabilités des résultats possibles d'une action n'est pas trivial et il nécessite que l'agent ait un modèle causal de l'environnement. Même si la fonction utilité est plus utile pour définir la performance d'un agent, il n'est pas toujours facile de déterminer l'utilité des divers états, surtout dans le cas où l'agent doit bâtir des plans compliqués d'actions.

2. Architectures des agents intelligents :

2.1. Définition :

Avant de commencer à présenter des architectures, il faut définir ce qu'on entend par architecture d'un agent. Même s'il n'y a pas consensus sur ce point, on peut dire que **l'architecture d'un agent** est une description de son organisation interne : les données et les connaissances de l'agent, les opérations qui peuvent être effectuées sur ses composantes et le flux de contrôle des opérations.
Le choix d'une architecture ou d'une autre est, bien sûr, lié à la structure conceptuelle de l'agent, décrite dans la section précédente, et représente la décision du concepteur sur la façon de bâtir l'agent artificiel. Les figures de la section précédente, qui décrivent la structure des divers types d'agents, sont des exemples d'architecture d'agents, bien qu'à un niveau de description très général. Nous allons voir dans ce qui suit des exemples d'architectures d'agents plus détaillées.

2.2. Architecture BDI :

Une architecture BDI est conçue en partant du modèle **"Croyance – Désir - Intention"**, en anglais "Belief-Desire-Intention", de la rationalité d'un agent intelligent. Dans ce qui suit, on va présenter d'une manière informelle, intuitive, la signification de ces trois éléments dans un modèle BDI. Le modèle a une théorie logique formelle associée, mais on ne va pas entrer dans les détails de celle-ci.

Le B = Belief = **Croyance**
Les croyances d'un agent sont les informations que l'agent possède sur l'environnement et sur d'autres agents qui existent dans le même environnement. Les croyances peuvent être incorrectes, incomplètes ou incertaines et, à cause de cela, elles sont différentes des connaissances de l'agent, qui sont des informations toujours vraies. Les croyances peuvent changer au fur et à mesure que l'agent, par sa capacité de perception ou par l'interaction avec d'autres agents, recueille plus d'information.

Le D = Desire = **Désir**
Les désirs d'un agent représentent les états de l'environnement, et parfois de lui-même, que l'agent aimerait voir réalisés. Un agent peut avoir des désirs contradictoires ; dans ce cas, il doit choisir parmi ses désirs un sous-ensemble qui soit consistant. Ce sous-ensemble consistant de ses désirs est parfois identifié avec les buts de l'agent.

Le I = Intention = **Intention**
Les intentions d'un agent sont les désirs que l'agent a décidé d'accomplir ou les actions qu'il a décidé de faire pour accomplir ses désirs. Même si tous les désirs d'un agent sont consistants, l'agent peut ne pas être capable d'accomplir tous ses désirs à la fois.

L'exemple suivant va apporter plus de clarification à ce modèle. L'agent Pierre a la croyance que, si quelqu'un passe son temps à étudier, cette personne peut faire une thèse de doctorat. En plus, Pierre a le désir de voyager beaucoup, de faire une thèse de doctorat et d'obtenir un poste d'assistant à l'université. Le désir de voyager beaucoup n'est pas consistant avec les deux autres et Pierre, après réflexion, décide de choisir, parmi ces désirs inconsistants, les deux derniers. Comme il se rend compte qu'il ne peut pas réaliser ses deux désirs à la fois, il décide de faire d'abord une thèse de doctorat. En ce moment Pierre a l'intention de faire une thèse et,

normalement, il va utiliser tous ses moyens pour y parvenir. Il serait irrationnel de la part de Pierre, une fois sa décision prise, d'utiliser son temps et son énergie, notamment ses moyens, pour voyager autour du monde. En fixant ces intentions, Pierre a moins de choix à considérer car il a renoncé à faire le tour des agences de voyage pour trouver l'offre de voyage qui le satisferait au mieux.

C'est cette idée même qui est au cœur de la **théorie BDI de l'action rationnelle**, proposée pour la première fois par Michael Bratman. C'est une théorie du raisonnement pratique qui essaie de comprendre comment les gens raisonnent dans la vie de tous les jours, en décidant, à chaque moment, ce qu'ils ont à faire. En développant sa théorie, Bratman montre que les intentions jouent un rôle fondamental dans le raisonnement pratique, car elles limitent les choix possibles qu'un humain (ou un agent artificiel) peut faire à un certain moment.

Une **architecture BDI** est alors un bon candidat pour modéliser le comportement d'un agent intelligent car :

- elle s'appuie sur une théorie connue et appréciée de l'action rationnelle des humains ;
- la théorie a été formalisée dans une logique symbolique formelle, rigoureuse ;
- elle a été implémentée et utilisée avec succès dans beaucoup d'applications.

La figure suivante présente les composantes principales d'une architecture BDI.

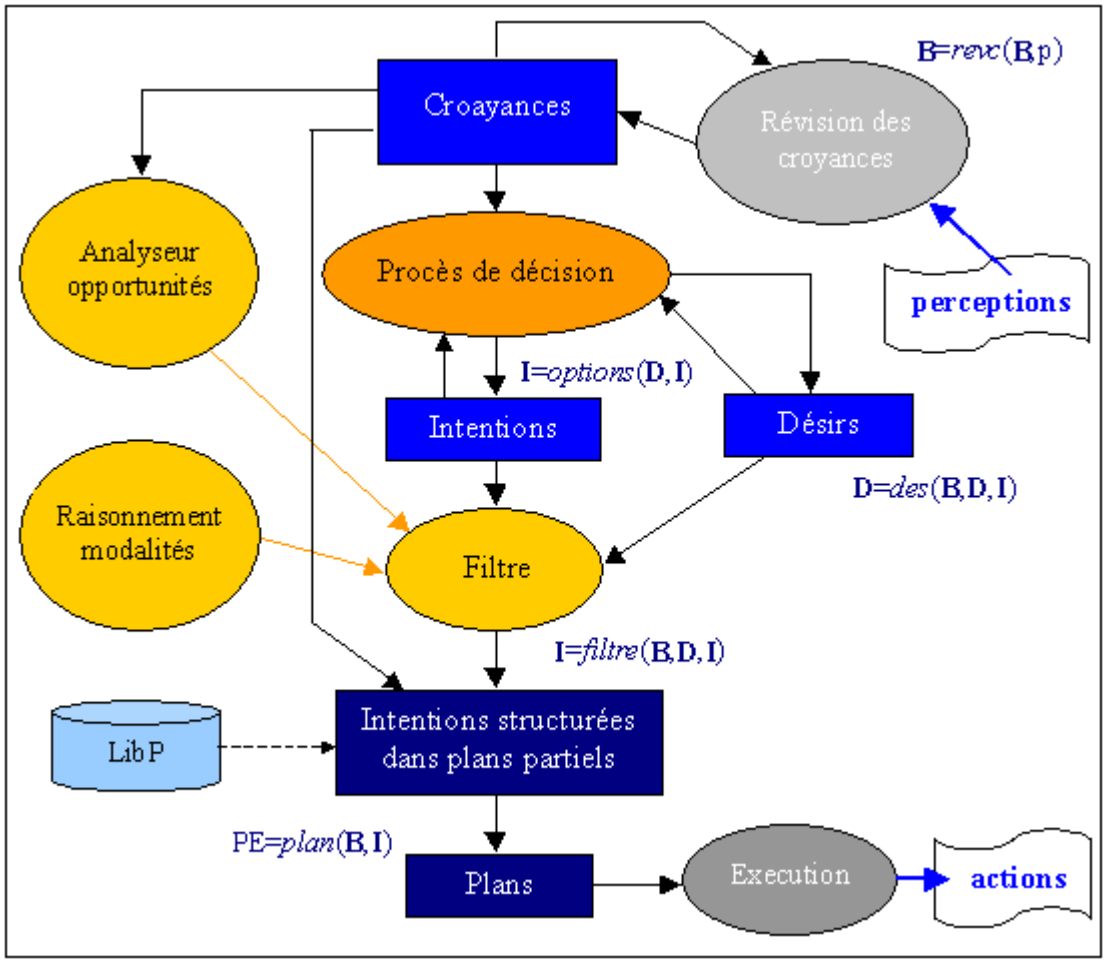

Figure 5 - Schéma d'une architecture BDI

L'agent a une représentation explicite de ses croyances, désirs et intentions. On dénote par B l'ensemble des croyances de l'agent, par D l'ensemble de ses désirs, et par I l'ensemble de ses intentions. Les ensembles B, D et I peuvent être représentés au moyen de divers modèles de représentation de connaissances, par exemple en utilisant la logique des prédicats du premier ordre, une logique d'ordre supérieur, le modèle des règles de production, ou bien comme de simples structures de données.

L'agent doit produire des plans, notamment une séquence d'actions qu'il va exécuter pour résoudre le problème. La représentation des actions la plus commune consiste à représenter les effets de ces actions sur l'environnement. Par exemple, si dans un état de l'environnement, l'agent déplace un chariot de la pièce A à la pièce B, la représentation de cette action sera "déplacer-chariot de A à B" et l'effet sur l'environnement, contiendra notamment le fait que le chariot n'est plus dans la pièce A et qu'il est dans la pièce B. Le résultat d'une action sera celui envisagé si l'environnement est déterministe. Les plans conçus par l'agent sont toujours des structures de connaissances ou structures de données. **LibP** est une bibliothèque de plans. Cette composante peut être présente ou non dans l'architecture. Si elle existe, le processus de planification de l'agent est simplifié car il peut retrouver des plans adéquats à une certaine situation en parcourant cette bibliothèque de plans. Dans la figure, les carrés représentent des structures de connaissances ou de données alors que les ovales représentent des composantes de contrôle et d'exécution.

Par la suite, nous considèrerons les fonctions suivantes :

- ***revc*** **:** B **x P ->**B est la fonction de révision des croyances de l'agent lorsqu'il reçoit de nouvelles perceptions sur l'environnement, où **P** représente l'ensemble des perceptions de l'agent ; elle est réalisée par la composante **Révision des croyances** ;
- ***options*** **:** D **x** I->I est la fonction qui représente le processus de décision de l'agent prenant en compte ses désirs et ses intentions courantes ; cette fonction est réalisée par la composante **Processus de décision** ;
- ***des*** **:** B **x** D **x** I->D est la fonction qui peut changer les désirs d'un agent si ses croyances ou intentions changent, pour maintenir la consistance des désirs de l'agent (on suppose dans notre modèle que l'agent a toujours des désirs consistants) ; cette fonction est également réalisée par la composante **Processus de décision** ;
- ***filtre*** **:**B **x** D **x** I->I est la fonction la plus importante car elle décide des intentions à poursuivre ; elle est réalisée par la composante **Filtre**.

La composante **Filtre** est la partie de l'architecture qui a la responsabilité de bâtir des plans partiels pour réaliser les intentions de l'agent, tout en tenant compte des nouvelles opportunités. En conséquence de ce qu'il perçoit de son environnement et de sa révision des croyances, l'agent peut détecter des nouvelles opportunités qui favorisent la réalisation de ses intentions ou qui peuvent même empêcher cette réalisation. Cette analyse est effectuée par la composante **Analyseur opportunités**. Ces nouvelles opportunités sont communiquées au **Filtre**. Le **Filtre** construit des plans partiels pour aboutir aux intentions de l'agent avec l'aide de la composante **Raisonnement modalités** ; cette dernière a la responsabilité d'effectuer le raisonnement orienté action et la modalité de réalisation des plans.

- ***plan*** **:** B **x** I-> **PE** est la fonction qui transforme les plans partiels en plans exécutables, **PE** étant l'ensemble de ces plans ; elle peut utiliser, par exemple, une bibliothèque de plans, représentée par le module **LibP** dans la figure.

Un **plan** est une séquence d'actions à exécuter dans le temps. Un plan partiel est un plan dans lequel tous les détails de planification n'ont pas été spécifiés, par exemple on s'est contenté d'un ordre partiel des actions dans le temps, ou certaines actions ne sont pas entièrement

détaillées. Prenons l'exemple suivant : on sait que pour partir en voyage, il faut faire ses valises et téléphoner à un ami pour dire au revoir, mais on n'indique ni dans quel ordre il faut faire ces deux actions, ni le nombre de valises à emporter. Ceci est un plan partiel. Une fois qu'on a décidé que l'on préparerait d'abord deux valises et que l'on appellerait ensuite son ami, on a raffiné le plan partiel, et obtenu ainsi un plan complètement spécifié et exécutable. La raison pour commencer par bâtir des plans partiels est que l'agent peut parfois être obligé de changer ses plans en fonction de nouvelles perceptions recueillies sur l'environnement, ainsi qu'il a été dit dans la section précédente. L'approche des plans partiels peut apporter une solution à ce problème. Une fois qu'on dispose d'un plan exécutable, le module **Exécution** va exécuter, l'une après l'autre, les actions de ce plan dans l'environnement.

2.3. Architecture réactive :

Les architectures réactives représentent le fonctionnement de l'agent au moyen de composantes avec une structure de contrôle simple, et sans représentation évoluée des connaissances de l'agent. L'intelligence de l'agent est vue comme étant le résultat des interactions entre ces composantes et l'environnement. Cela veut dire qu'une telle architecture peut résoudre des problèmes complexes, qui normalement demandent un comportement intelligent, sans traiter l'intelligence du point de vue classique de l'intelligence artificielle. On dit que l'intelligence émerge de l'interaction entre des composantes simples, et entre les agents réactifs et l'environnement. Cette approche, qui diffère beaucoup de la conception des agents intelligents, est apparue comme une solution aux critiques visant les approches symboliques, notamment la complexité des calculs nécessités par ces approches, qui paraît incompatible avec les ressources limitées des agents, et la difficulté de trouver toujours le bon modèle cognitif pour certaines applications.

L'architecture réactive la plus connue et la plus influente est celle proposée par Rodney Brooks ; elle s'appelle **architecture de subsomption**, en anglais "subsumption architecture". Une architecture de subsomption comporte plusieurs modules, chaque module étant responsable de la réalisation d'une tâche simple. Ces modules correspondent à des comportements spécifiques pour accomplir une tâche particulière, et s'appellent **modules de compétence**. On peut voir cette architecture, dans un premier temps, comme ayant la structure des agents, où l'agent a plusieurs composantes de décision, chaque composante correspondant à un module de l'architecture de subsomption. Pour la perception de l'environnement, plusieurs modules peuvent assumer l'exécution d'une action différente ; pour choisir l'action la plus opportune, les modules sont organisés en couches hiérarchisées, chaque couche ayant une priorité différente. Les couches supérieures correspondent à des tâches plus abstraites qui sont détaillées à l'aide des tâches plus concrètes et plus simples, les couches supérieures ayant une priorité plus petite que les couches inférieures. Les couches inférieures correspondent aux tâches simples et elles ont une priorité plus grande.

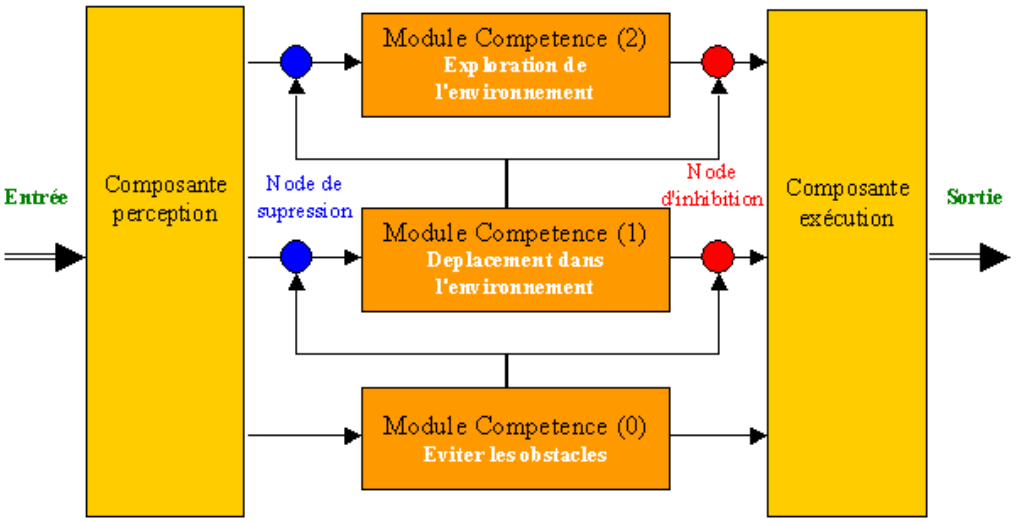

Figure 6 - Schéma d'une architecture réactive de subsomption

Si on utilise cette architecture pour construire un robot qui doit faire l'exploration de la planète Mars, on peut définir :

- M0 - un module qui a la compétence d'éviter les obstacles ;
- M1 - un module M1 qui est responsable des déplacements dans l'environnement tout en évitant les obstacles à l'aide de M0 ;
- M2 - un module qui a la compétence supérieure, la plus abstraite, de faire l'exploration systématique de la planète en se déplaçant grâce aux actions du module M1.

Un module sur une couche inférieure a une priorité plus grande qu'un module situé sur une couche plus élevée, parce qu'il est responsable d'une tâche plus simple mais plus "urgente". Dans ce but, le fonctionnement d'un module situé sur une couche supérieure est subordonné à un module inférieur. Un module sur une couche inférieure peut modifier l'entrée d'un module supérieur au moyen d'un nœud de suppression, et invalider l'action du module supérieur au moyen d'un nœud d'inhibition, comme on peut le voir dans la figure 5. Par exemple, si notre robot veut se déplacer vers l'Est en partant d'une certaine position et qu'il n'y a pas d'obstacle dans cette direction, l'action exécutée par la composante exécution est celle commandée par M1 de se déplacer vers l'Est. Si, par contre, il y a un obstacle, le module M0 prend en compte cet obstacle par sa perception de l'environnement et inhibe le déplacement vers l'Est. M1 essaiera alors de se déplacer dans une autre direction. C'est cette organisation qui justifie l'appellation de subsomption de l'architecture.

Le fonctionnement de l'agent est décrit par un ensemble de règles de comportement, "behaviour rules" en anglais. Une **règle de comportement** est semblable à une règle de production et elle a deux parties : une condition **c** et une action **a**. La condition correspond à une perception de l'environnement, et l'action à une action possible d'un module de compétence. Soit

- **Comp** = {(**c**, **a**) | **c dans P**, **a dans A**} l'ensemble de règles de comportement, où **P** désigne l'ensemble des perceptions, et **A** les actions possibles de l'agent. à chaque module de compétence est associé un sous-ensemble des règles de comportement spécifiques à ses compétences.

On peut définir une **relation d'ordre** ou relation inhibitrice totale sur l'ensemble **Comp**, **Đ**: **Comp** x **Comp** qui établit la priorité des modules de compétence. Soit **R** l'ensemble des règles de comportement qui peuvent être exécutées à un certain moment, notamment les règles (**c**, **a**) pour lesquelles la perception **p** sur l'environnement satisfait la condition **c**. Alors, les règles qui sont effectivement exécutées sont :

{(**c**, **a**)| {(**c**, **a**) **dans R** et non existe pas (**c'**, **a'**) **dans R** tel que (**c'**, **a'**) < (**c**, **a**)}

De cette manière, on peut indiquer quel module aura la décision de l'action à exécuter pour une certaine perception sur l'environnement.

Les architectures réactives ont l'avantage de la simplicité et de l'efficacité de calcul. Pourtant, elles présentent plusieurs limitations, ce qui fait que ces architectures ne peuvent pas être utilisées dans de nombreuses classes d'applications. Les principales objections contre les architectures réactives sont les suivantes : les agents ont une vision de courte durée sur la résolution du problème, et ils peuvent ne pas toujours choisir la meilleure action à exécuter à un certain moment ; comme les agents réactifs ne maintiennent pas une représentation de l'environnement, ils ne peuvent pas avoir des buts, et encore moins doter les états d'utilité ; si on a besoin de beaucoup de couches pour modéliser le comportement de l'agent, on ne peut pas toujours prévoir toutes les interactions possibles ; dans ce cas, même le grand avantage de cette architecture, l'émergence de l'intelligence par interaction, peut devenir un désavantage, car des interactions indésirables peuvent émerger. Pour avoir le meilleur des deux solutions, à savoir architectures cognitives et réactives, les chercheurs ont conçu des architectures hybrides qui combinent les caractéristiques de ces deux architectures.

2.4. Architecture hybride :

Une architecture hybride d'un agent intelligent est une architecture composée d'un ensemble de modules organisés dans une hiérarchie, chaque module étant soit une composante cognitive avec représentation symbolique des connaissances et capacités de raisonnement, soit une composante réactive. De cette manière, on combine le comportement pro-actif de l'agent, dirigé par les buts, avec un comportement réactif aux changements de l'environnement. En plus, on espère obtenir simultanément les avantages des architectures cognitives et réactives, tout en éliminant leurs limitations.

Plusieurs architectures hybrides ont été conçues et utilisées. Une des architectures hybrides les plus connues est celle du **système InteRRaP** ("Integration of Reactive Behavior and Rational Planning" = Intégration du comportement réactif et planification rationnelle). L'architecture InteRRaP est une architecture en couches avec des **couches verticales** où les données d'entrée, notamment les perceptions, passent d'une couche à l'autre, comme on le voit sur la figure ci-dessous. En cela, elle est un peu différente de l'architecture de subsomption qui est une architecture en **couches horizontales**, où les perceptions sont transmises directement à toutes les couches à la fois.

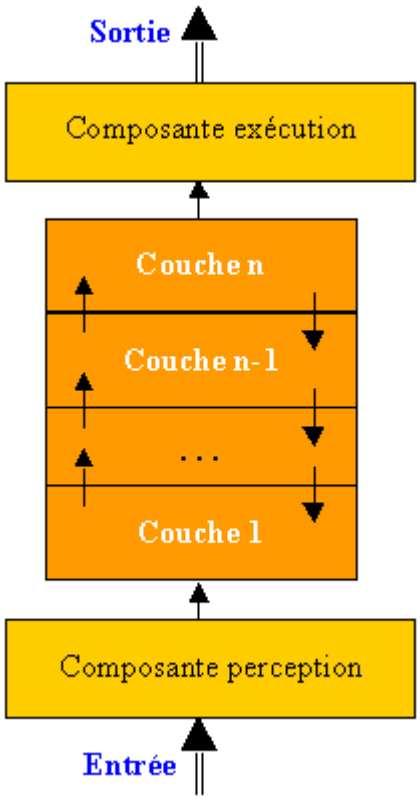

Figure 7 - Architecture hybride en couches verticales

Un agent InteRRaP est un agent BDI qui a des buts à atteindre (les buts sont les mêmes que les désirs) et qui est capable de coopérer avec d'autres agents InteRRaP pour accomplir ces buts. Les buts d'un agent sont divisés en trois catégories :

- **réactions** : ce sont des buts simples à accomplir en fonction des perceptions sur l'environnement ;
- **buts locaux** : ce sont des buts que l'agent peut accomplir par lui-même ;
- **buts coopératifs** : ce sont les buts qui peuvent être accomplis uniquement par une coopération avec d'autres agents dans le système.

Il est intéressant d'observer que, dans cette conception, la réactivité de l'agent est conçue toujours comme un but, c'est-à-dire au niveau cognitif, mais comme un but très simple à réaliser.

L'architecture InteRRaP est composée de **trois couches de contrôle** et **trois bases de connaissances** associées qui représentent l'agent et l'environnement à divers niveaux d'abstraction, comme il est indiqué sur la figure ci-dessous.

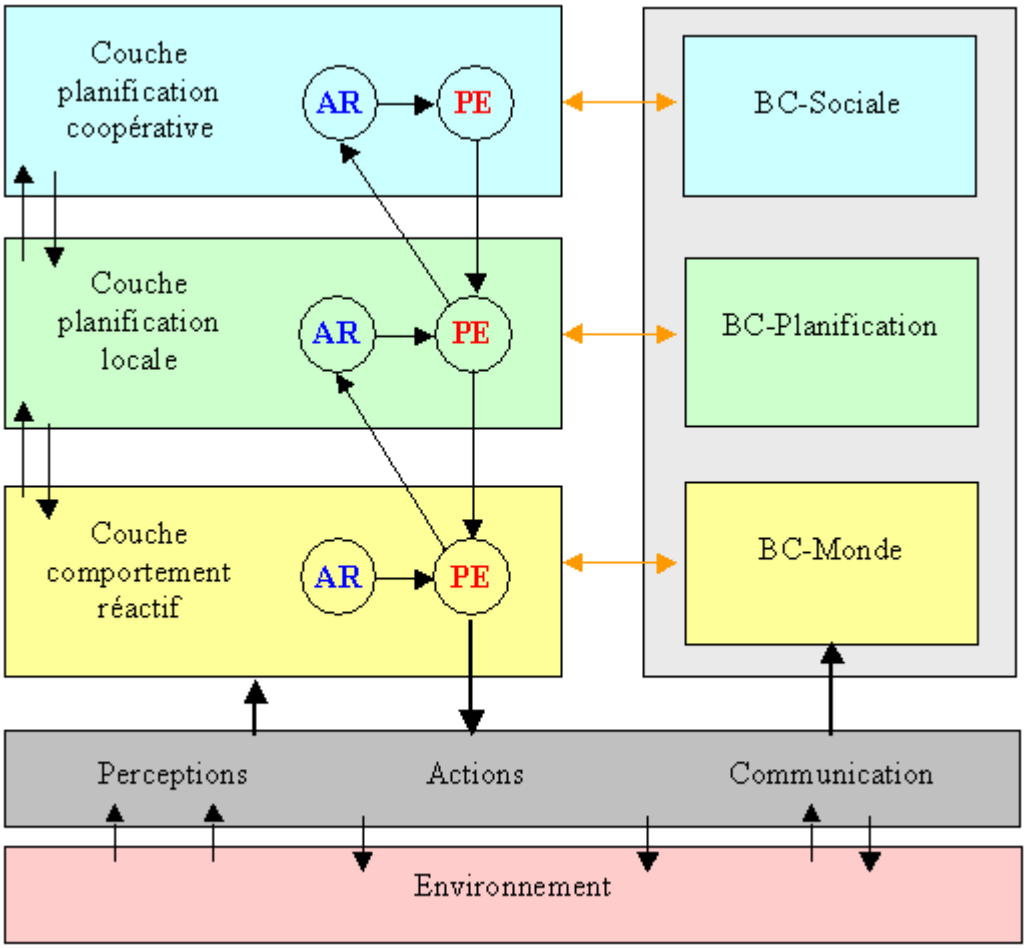

Figure 8 - Architecture InteRRaP

Chaque couche a un ensemble d'opérations spécifiques associées et une couche supérieure utilise les opérations plus simples de la couche d'au-dessus pour exécuter ses opérations plus élaborées. Le flux de contrôle passe de bas en haut, et une couche prend le contrôle lorsque la couche antérieure ne peut plus contribuer, par ses opérations, à l'accomplissement des buts. Chaque couche comprend deux modules: un **module pour l'activation** des buts et la reconnaissance des situations (**AR**) et un **module de planification et d'exécution** (**PE**). Les perceptions sur l'environnement sont transmises au module **AR** de la première couche et, de module en module, vers le sommet de la hiérarchie. Le flux de contrôle des actions passe de haut en bas pour arriver à la fin au module **PE** de la dernière couche, et les actions associées sont exécutées sur l'environnement.

La base de connaissances **BC-Monde** représente l'information que l'agent possède sur l'environnement (croyances sur l'environnement), la base de connaissances **BC-Planification** est équivalente à la bibliothèque des plans d'une architecture BDI ; enfin, **BC-Sociale** représente les croyances de l'agent sur les autres agents du système, et notamment leurs capacités de l'aider à atteindre ses buts.

La figure ci-dessous présente la relation entre les modules **AR** et **PE** et les croyances, les buts, les intentions et les plans de l'agent. On a indiqué sur la figure les fonctions BDI correspondant aux divers modules.

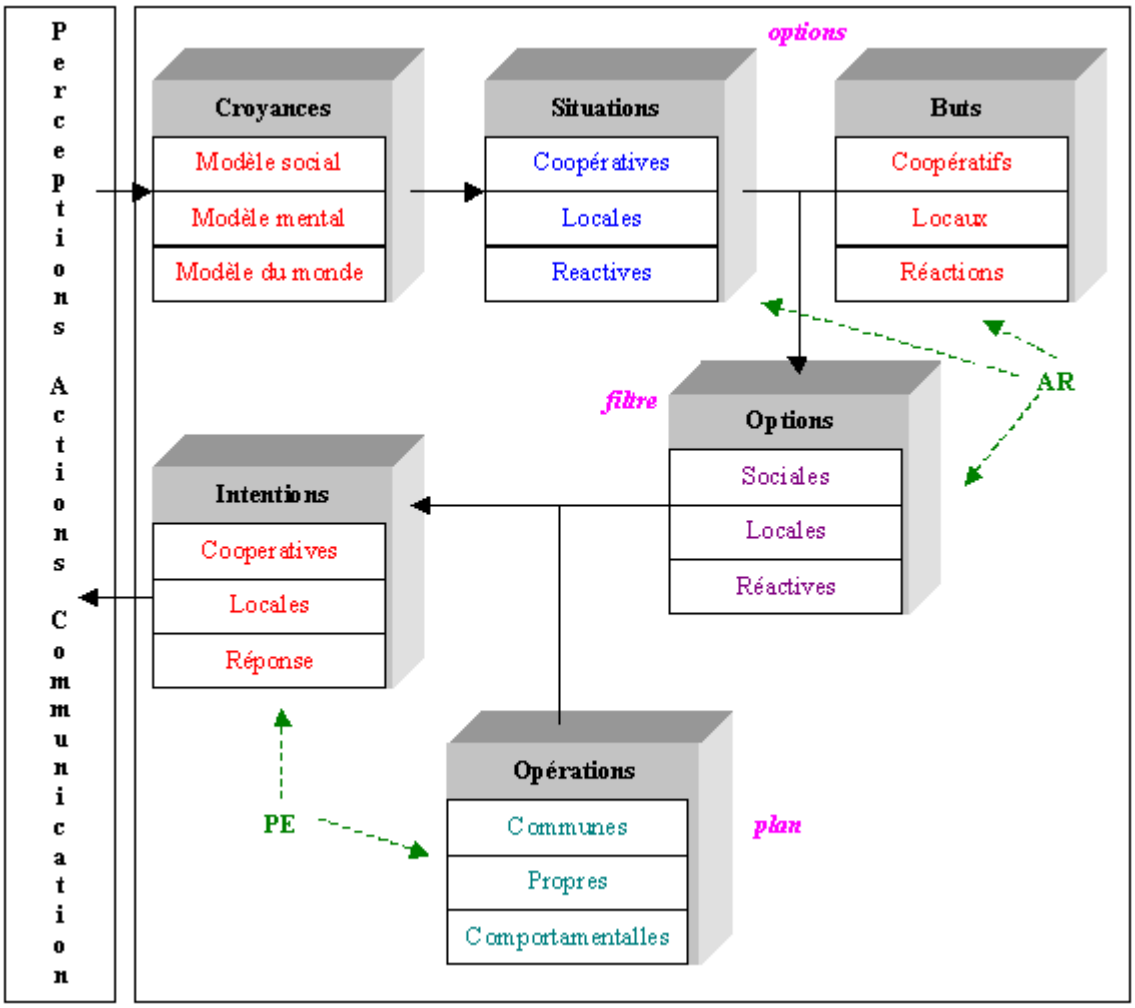

Figure 9 – Le modèle BDI dans l'architecture InteRRaP

Il faut noter que, par rapport aux autres architectures discutées précédemment, l'architecture InteRRaP comprend une représentation explicite du processus de coopération d'un agent avec d'autres agents du système. Dans les chapitres à venir nous allons voir d'autres exemples d'architectures où cette coopération est représentée de manière explicite.
Une autre architecture hybride bien connue est l'architecture du système TuringMachine. Cette architecture est une architecture en couches horizontales, semblable à l'organisation de l'architecture de subsomption, où chaque couche est responsable de l'exécution d'actions de plus en plus complexes.

Les architectures hybrides se sont avérées efficaces dans beaucoup d'applications. Il y a cependant deux reproches qu'on peut leur faire : il n'y a pas de modèle formel associé comme dans le cas d'une architecture BDI pure, et on n'a pas encore validé de véritable méthodologie pour guider l'utilisateur dans la conception d'applications utilisant de telles architectures.

3 – Les systèmes multi-agents

1. Agents et systèmes multi-agents :

Dans les généralités, on a distingué deux catégories d'agents : Les agents réactifs et les agents cognitifs. Ces agents sont utilisés pour la création de systèmes multi-agents.

Les agents réactifs

Un SMA constitué d'agents réactifs possède généralement un grand nombre d'agents et présente un comportement global intelligent. Les agents réactifs sont considérés intelligents au niveau du groupe, du système. En conséquence, l'intelligence est distribuée entre beaucoup d'agents réactifs et le comportement intelligent devrait émerger de l'interaction entre ces agents réactifs et l'environnement.
Cependant, la convergence du comportement de l'ensemble des Agents vers un état stable n'est pas forcément assurée, et si un état stable est atteint, il n'est pas certain qu'il s'agisse de la solution optimale.

Les agents cognitifs

Un SMA constitué d'agents cognitifs possède communément peu d'agents. Ils réclament des ressources plus importantes que les systèmes d'agents réactifs.
La convergence du système vers un état décisionnel stable n'est pas non plus assurée par l'utilisation de ce type d'Agents, mais ils permettent de résoudre des problèmes plus complexes et nécessitant une plus grande abstraction.

2. Interactions entre agents :

2.1. Définition d'une interaction :

Une des principales propriétés de l'agent dans un SMA est celle d'interagir avec les autres agents. Ces interactions sont généralement définies comme toute forme d'action exécutée au sein du système d'agents et qui a pour effet de modifier le comportement d'un autre agent.
Elles permettent aux agents de participer à la satisfaction d'un but global. Cette participation permet au système d'évoluer vers un de ses objectifs et d'avoir un comportement intelligent indépendamment du degré de complexité des agents qui le composent.

En général, les interactions sont mises en œuvre par un transfert d'informations entre agents ou entre l'environnement et les agents, soit par perception, soit par communication :

- par **la perception**, les agents ont connaissance d'un changement de comportement d'un tiers au travers du milieu.
- par **la communication**, un agent fait un acte délibéré de transfert d'informations vers un ou plusieurs autres agents.

L'interaction peut être décomposée en trois phases non nécessairement séquentielles :

- la réception d'informations ou la perception d'un changement,
- le raisonnement sur les autres agents à partir des informations acquises,
- une émission de message(s) ou plusieurs actions (plan d'actions) modifiant l'environnement. Cette phase est le résultat d'un raisonnement de l'agent sur son propre savoir-faire et celui des autres agents.

Le degré de complexité des connaissances nécessaires pour traiter les interactions dépend des capacités cognitives (de raisonnement) de l'agent et du fait que l'agent ait connaissance ou non de l'objectif du système global.
En effet, un agent qui poursuit un objectif individuel au sein du système, comme c'est le cas pour les agents dits réactifs, ne focalise pas son énergie pour interagir avec les autres même s'il y est amené. Par contre, un agent qui participe à la satisfaction du but global du système tout en poursuivant un objectif individuel, va passer une partie de son temps à coopérer ou à se coordonner avec les autres agents. Pour cela, il doit posséder des connaissances sociales qui modélisent ses croyances sur les autres agents.

2.2. Situations d'interaction :

L'interaction constitue un niveau d'abstraction supérieur à la notion de communication et d'action. Le problème est alors de savoir combiner ces éléments (communications et actions) afin de coordonner et de contrôler les échanges entre plusieurs agents pour avoir un comportement collectif cohérent du système.

Plusieurs types d'interaction ont été définis et analysés à travers divers composantes (Ferber, 1995). Ferber donne une classification des situations d'interaction abordée selon le point de vue d'un observateur extérieur. Cette classification présente les différentes situations d'interaction que l'on retrouve en fonction des objectifs des agents (compatibles ou incompatibles), des ressources dont ils disposent (suffisantes ou insuffisantes) et de leurs compétences pour la résolution d'un problème.

Buts	Ressources	Compétences	Situation
Compatibles	Suffisantes	Suffisantes	Indépendance
Compatibles	Suffisantes	Insuffisantes	Collaboration simple
Compatibles	Insuffisantes	Suffisantes	Encombrement
Compatibles	Insuffisantes	Insuffisantes	Collaboration coordonnée
Incompatibles	Suffisantes	Suffisantes	Compétition individuelle pure
Incompatibles	Suffisantes	Insuffisantes	Compétition collective pure
Incompatibles	Insuffisantes	Suffisantes	Conflits individuels pour des ressources
Incompatibles	Insuffisantes	Insuffisantes	Conflits collectifs pour des ressources

Figure 10 - Tableau des différentes situations d'interactions

On aboutit ainsi à une première typologie des situations d'interactions : indépendance, collaboration simple, encombrement, collaboration coordonnée, compétition individuelle pure, compétition collective pure, conflit individuel, conflits collectifs.

Cette classification permet de distinguer des situations d'interaction. En revanche, elle ne propose pas de moyens aux agents pour détecter le type de situation dans laquelle ils se trouvent ainsi que leur besoin de coopérer.

Ces différentes situations d'interaction peuvent être affinées en décrivant d'autres paramètres beaucoup plus détaillés tels que la reconnaissance des buts communs, les désirs, les croyances, etc. Les situations d'interaction peuvent ainsi être analysées de manière hiérarchique. En effet, une situation d'interaction complexe est composée de situations plus élémentaires. On peut ainsi distinguer des macro-situations d'interaction qui sont caractéristiques d'une analyse globale de l'activité d'un ensemble d'agents et des micro-situations qui se situent à un niveau de détail plus fin.

Il est possible de différencier les comportements selon les situations observés habituellement dans les environnements multi-agents (les situations routinières, les situations familières, et les situations non-familières (ou de crise)) grâce à un modèle qui intègre à la fois le niveau réactif et le niveau cognitif d'un agent. Le comportement d'un agent face à différentes situations de décision se résume par les phases suivantes : *Tout d'abord, l'agent perçoit une ou plusieurs informations venant de son environnement. Cette information le pousse soit à agir si elle est directement perçue sous forme d'action, soit à planifier si elle est perçue comme une tâche ou un but. Si en revanche, l'information n'est perçue sous aucune de ces deux formes, l'agent doit alors identifier ou reconnaître la situation relative à l'information perçue qui peut être identifié en terme d'action ou de but. Enfin, si l'agent est face à une ambiguïté (situation inhabituelle), alors il doit prendre des décisions qui doivent l'amener à s'engager dans une action ou un but donné.*

Les situations citées plus haut ne peuvent être considérées comme des situations d'interaction car elles sont locales aux agents (une situation telle qu'elle est définie est propre à un agent et peut être perçue différemment par d'autres agents). Ces situations permettent donc de définir le comportement d'un agent et son évolution dans son environnement.

3. Communications :

3.1. Définition de la communication :

Les communications, dans les SMA comme chez les humains, sont à la base des interactions et de l'organisation. Une communication peut être définie comme une forme d'action locale d'un agent vers d'autres agents. Les questions abordées par un modèle de communication peuvent être résumées par l'interrogation suivante : qui communique quoi, à qui, quand, pourquoi, et comment ?

- ***Pourquoi les agents communiquent-ils*** ? La communication doit permettre la mise en œuvre de l'interaction et par conséquent la coopération et la coordination d'actions.
- ***Quand les agents communiquent-ils*** ? Les agents sont souvent confrontés à des situations où ils ont besoin d'interagir avec d'autres agents pour atteindre leurs buts locaux ou globaux. La difficulté réside dans l'identification de ces situations. Par exemple, une communication peut être sollicitée suite à une demande explicite par un autre agent.
- ***Avec qui les agents communiquent-ils*** ? les communications peuvent être sélectives sur un nombre restreint d'agents ou diffusées à l'ensemble des agents. Le choix de l'interlocuteur dépend essentiellement des accointances de l'agent (connaissances qu'a l'agent sur les autres agents).
- ***Comment les agents communiquent-ils*** ? La mise en œuvre de la communication nécessite un langage de communication compréhensible et commun à tous les agents. Il faut identifier les différents types de communication et définir les moyens permettant non seulement l'envoi et la réception de données mais aussi le transfert de connaissances avec une sémantique appropriée à chaque type de message.

3.2 Langages de communication inter-agents :

Grâce à la coordination un système multi-agents peut réaliser ses tâches avec plus d'efficacité qu'un seul agent. Mais pour coordonner l'activité d'un ensemble hétérogène d'agents autonomes, il faut que les agents communiquent dans un langage compréhensible par tous les autres. On observe que dans un système ouvert un tel langage peut constituer une interface entre les agents.

L'utilisation d'un langage commun implique que tous les agents comprennent son vocabulaire sous tous ses aspects concernant:

- **la syntaxe**, qui précise le mode de structuration des symboles;
- **la pragmatique** pour pouvoir interpréter les symboles;
- **l'ontologie** pour pouvoir utiliser les mêmes mots d'un vocabulaire commun.

La compréhension du sens des symboles, ou à quoi les symboles font référence, demande un standard rigoureux de la sémantique et de **la pragmatique**. De plus il est nécessaire que les agents sachent bien utiliser le vocabulaire pour atteindre leurs buts, éviter les conflits, coopérer pour exécuter leurs tâches et modifier l'état mental d'un autre agent.

Un Langage de Communication Agent (ACL de l'anglais Agent Communication Language) doit être conçu comme un langage de haut niveau qui assure en premier lieu l'échange d'états mentaux et le sens du vocabulaire [BOISSIER]. Le format utilisé pour l'échange des

connaissances est donné par un langage de contenu, indépendant du langage ACL (p.ex.. KIF, FIPA-SL, FIPA-CCL). Le vocabulaire commun concerne les définitions précisées dans une ontologie. Ces composants sont représentés dans la figure 11.

Figure 11 - Modèle des Langages de Communication entre Agents

Les aspects techniques d'implantation d'un ACL concernent l'existence dans le système de communication entre les agents des mécanismes ci-dessous:

- des protocoles pour la couche de transport utilisé (TCP/IP, UDTP, SMTP, IIOP, HTTP)
- des protocoles de haut niveau (e.g. contract-net, licitations ('auctions'), enregistrement des noms)
- des services d'infrastructure (broker, facilitateurs, loggers etc.)
- un mécanisme de contrôle de la communication au sein des agent

Les échanges d'information peuvent être faits par:

- messages (point à point, diffusion, synchrone ou asynchrone)
- mémoire partagée (tableaux noirs)

Le développement des spécifications des ACL's tire profit des recherches effectuées sur les langues naturelles, sur la pragmatique conversationnelle et la théorie des actes de langage.

L'intérêt des langages d'interaction entre agents est de réduire les communications en évitant une description exhaustive des messages *ad hoc* et une gamme inutilement étendue de protocoles. Ces langages se focalisent essentiellement sur la manière de décrire exhaustivement des actes de communication d'un point de vue syntaxique et sémantique supportant un langage de représentation des connaissances. Toutefois, l'aspect ontologique et l'utilisation de conventions garantissant un comportement collectif cohérent du système et l'aspect conversationnel n'est pas facile à garantir. Plusieurs tentatives de normalisation de la communication inter-agents ont été effectuées au sein de la communauté multi-agents ces dernières années.

3.3. Langage de communication KQML :

Aux Etats-Unis, un standard de fait de langage de communication de haut niveau appelé KQML “ Knowledge Query and Manipulation Language ” (Labrou et Finin, 1997) a été développé. Ce dernier est fondé sur la théorie des actes de langage dans le but de permettre aux agents cognitifs de coopérer. Le contenu du message échangé est une expression spécifiée

en KIF (Knowledge Interchange Format) qui utilise le formalisme de la logique de premier ordre.

KQML est un langage de communication et un protocole de haut niveau pour l'échange de l'information, orienté messages et indépendant de la syntaxe du contenu et de l'ontologie applicable.

En plus, KQML est indépendant du mécanisme de transport (TCP/IP, SMTP, IIOP ou autre), indépendant du langage du contenu échangé (KIF, SQL, Prolog ou autre) et indépendant de l'ontologie utilisé.

Conceptuellement, nous pouvons identifier trois couches dans un message de KQML : contenu, communication et message.

- la couche " contenu " comporte la teneur réelle du message utilisant un langage de représentation propre au système. KQML peut supporter n'importe quel langage de représentation, y compris des langages exprimés en ASCII et ceux exprimés en utilisant la numération binaire.
- la couche de "communication " code un ensemble de dispositifs au message qui décrivent les paramètres de communication les plus bas, tels que l'identité de l'expéditeur et du destinataire et un identifiant unique associé à la communication.
- la couche " message ", qui code un message qu'une application voudrait transmettre à une autre, est le noyau de KQML. Cette couche détermine les genres d'interactions au sein des agents dialoguant via KQML. La fonction de la couche message est d'identifier l'acte du langage ou la performative que l'expéditeur attache au contenu. Cet acte de langage indique si le message est une affirmation, une question, une commande ou tout autre d'un ensemble de performatives connus (types de messages primitifs). En outre, puisque le contenu est opaque à KQML, le message inclut également les options qui décrivent le langage du contenu, l'ontologie qu'il suppose utiliser et une certaine description du contenu, tel qu'un descripteur appelant une matière dans l'ontologie. Ces dispositifs permettent une analyse correcte des messages sans avoir à accéder au contenu.

Les messages de KQML ne concernent pas seulement des phrases dans un langage quelconque, mais sont enrichis d'une attitude sur le contenu (affirmation, désengagement, requête, question, etc.).

Ce langage propose une description d'un nombre important de performatives permettant les conversations entre agents mais manque de spécifications et de formalisation (cf. les critiques de Cohen et Levesque, 1995).

On peut prendre comme exemple de message KQML, le message d'un agent E (émetteur) qui demande aux autres agents récepteurs (R) quel est le prix d'une imprimante à jet d'encre (la question est placée dans le contenu du message)

```
(ask-all
    :content (PRIX JET ?prix)
    :sender Ag1
    :reply-with prixJet
    :ontology imprimantes
    :language LPROLOG
   )
```

La *'performative'* est *ask-all* qui signifie que l'agent E désire que tous les agents R répondent à sa question. .

Les attributs des cette performative sont :

- content - le contenu du message (l'information transportée par la performative)
- reply-with - identificateur unique du message, en vue d'une référence ultérieur
- ontology - précise le nom de l'ontologie utilisé dans conten
- language - le nom du langage utilisé dans le contenu du message (content)

D'autres attributs, largement utilisés, sont:

- sender - identifie l'agent émetteur (le nom de l'agent qui envoie la performative)
- receiver - identifie le destinataire du message (nom de l'agent qui reçoit la performative)
- in-reply-to - référence à un message auquel l'agent est entrain de répondre (dans une réponse c'est le symbole précisé par l'attribut reply-with de l'émetteur)
- force - l'émetteur ne contredira jamais le contenu du message

3.4. Langage de communication FIPA-ACL :

Récemment, la collaboration internationale des membres d'organisations universitaires et industrielles regroupées au sein de FIPA (*Foundation for Intelligent Physical Agents*) a permis de spécifier des standards dans la technologie agent et vise à favoriser l'interopérabilité des applications, des services et des équipements informatiques basés sur le paradigme agent. Ils ont défini un certain nombre de spécifications principales d'agents. Notamment, un standard de langage de communication agent ACL (*Agent Communication Language*) (FIPA, 97) a été proposé et spécifié. Comme KQML, ce dernier est basé sur la théorie des actes de langage : les messages sont des actions ou des actes communicatifs, car ils sont prévus pour effectuer une certaine action en vertu de l'envoi. Les spécifications de FIPA-ACL se composent d'un ensemble de types de message et de la description de leur pragmatique qu'est, des effets sur les attitudes mentales des agents (expéditeur et récepteur). Les spécifications décrivent chaque acte communicatif avec une forme narrative et une sémantique formelle basés sur la logique modale.

Elles fournissent également la description normative d'un ensemble de protocoles d'interaction de haut niveau, y compris la demande d'action, l'établissement de contrat (contract net) et plusieurs genres de ventes aux enchères.

FIPA-ACL est superficiellement semblable à KQML. Sa syntaxe est identique à celle de KQML excepté différents noms pour quelques primitifs réservés. Ainsi, il maintient l'approche de KQML de distinguer le langage externe du langage interne. Le langage externe définit la signification prévue du message; le langage interne ou le contenu, dénote l'expression à laquelle s'appliquent les croyances, les désirs, et les intentions des interlocuteurs.

Les différentes approches développées à ce jour respectent partiellement les bases de la théorie des actes de langage (Searle, 1969). Elles considèrent que l'interaction est basée sur des actions communicatives mutuellement échangées entre les agents et qui sont initiées par des intentions, ce qui justifie l'utilisation d'une logique pour raisonner sur les intentions.

L'enchaînement des actes de communication de telle sorte à produire une "discussion" cohérente entre les agents nécessite une formalisation non seulement au niveau de ses

composantes, c'est-à-dire les actes de communication. La logique et la sémantique des actes doivent alors être étendues pour la problématique de l'enchaînement par la notion de protocole.

Les documents de référence de FIPA-ACL et de KQML ne spécifient aucune implantation par l'utilisation d'un langage de programmation ni de plate-forme particulière. Ce champ de développement est particulièrement propre à chaque équipe désirant implanter un tel langage. La seule obligation est que l'implantation doit être conforme aux spécifications du langage.
Toutefois, il existe plusieurs efforts d'implantation, dont les principaux sont JATLite, KAPI et COBALT proposé par l'équipe IRIT/SIERA de Toulouse.

4. Négociation dans les systèmes multi-agents :

Dans un système multi-agents les agents interagissent en vue de réaliser des tâches ou d'atteindre des buts. L'interaction a lieu, d'habitude dans un environnement commun où les agents ont diverses **zones d'influence**, notamment diverses parties de l'environnement sur lesquelles ils peuvent agir. Ces zones peuvent être disjointes mais, dans la plupart des cas, elles se superposent - l'environnement est partagé par les agents. Rappelez-vous aussi la classification des situations d'interactions possibles entre agents en fonction de leurs buts, ressources et compétences. En interagissant dans un tel environnement partagé, les agents doivent coordonner leurs actions et avoir des mécanismes pour la résolution des conflits. La coordination et la résolution des conflits sont surtout nécessaire dans le cas des agents egocentrés (des agents ayants leurs propres buts, désirs, préférences, etc.) ou compétitifs mais aussi bien, parfois, dans le cas des agents coopératifs pour la communication des changements des plans ou l'allocation des tâches.

Le mécanisme favori pour la **résolution des conflits et la coordination**, inspiré du modèle des humains, est **la négociation**.

Dans le cas des agents intelligents et dans les systèmes multi-agents, la négociation est une composante de base de l'interaction surtout parce que les agents sont autonomes (Jenning e.a. 2000); il n'y a pas de solution imposée à l'avance et les agents doivent arriver à trouver des solutions dynamiquement, pendant qu'ils résolvent les problèmes.

On peut citer plusieurs définitions pour la négociation. David et Smith (1980) disent que:
"Par négociation, on entend une discussion dans laquelle des individus intéressés échangent des informations et arrivent à un accord en commun."

Pruitts (1981) donne une définition qui s'appui sur des considérations psychologiques et pour laquelle le conflit est l'élément de base.
"La négociation est le processus par lequel plusieurs individus prennent une décision commune. Les participants expriment d'abord des demandes contradictoires, puis ils essaient de trouver un accord par concession ou par la recherche de nouvelles alternatives."

Si on regarde les deux définitions citées, on peut identifier deux aspects essentiels de la négociation: la communication et la prise de décisions.

Pour modéliser la négociation dans un logiciel multi-agents il faut alors prendre en compte les **aspects** suivants:

- **Le langage de négociation** - le langage utilisé par les agents pour échanger des informations pendant la négociation; le langage de négociation est composé d'un ensemble de primitives de communication.
- **Le protocole de négociation** - l'ensemble des règles qui régit la négociation: les participants possibles dans la négociation, les propositions légales que les participants peuvent faire, les états de la négociation (par exemple l'état initial où commence la négociation, l'état où on accepte des soumissions ou la fin de la négociation) et une règle pour déterminer quand on est arrivé à un accord ou quand il faut s'arrêter parce qu'aucun accord n'a pas n'a pu être trouvé .
- **L'objet de négociation** - un objet abstrait qui comprend les attributs qu'on veut négocier; dans certains cas il s'agit de négocier uniquement le prix, mais dans d'autres cas il faut aussi négocier plusieurs attributs comme le temps nécessaire pour satisfaire une commande, la qualité des produits, etc.

- **Le processus de décision** - le modèle que l'agent utilise pour prendre des décisions pendant la négociation. La partie la plus importante de la prise des décisions dans ce cas est la **stratégie de négociation** qui permet de déterminer quelle primitive de négociation l'agent doit choisir à un certain moment. Le processus de décision revient à répondre à la question: "Que dois-je faire maintenant?", par exemple liciter enchérir, abandonner, etc. Pour prendre une décision adéquate, un agent doit être capable de faire un raisonnement stratégique, notamment raisonner en tenant compte de ce que font/décident les autres agents et, s'il parvient à le savoir ou à le supposer, quel est le modèle de décision des autres agents.

Le protocole établit les règles de négociation. Par exemple, une négociation peut avoir lieu en un seul tour, comme l'enchère premier-prix offre-cachée, ou en plusieurs tours avec les participants faisant des offres (soumission) à chaque tour. L'exemple qui suit (qui peut avoir lieu aussi bien entre agents humains que logiciels) montre une négociation avec plusieurs tours.

> *Agent A:* Je te demande de trouver une compagnie aérienne qui offre des vols Paris-Bucarest et retour en dessous de 300 euros.
>
> *Agent B:* Oui, je peux le faire, pourvu que tu m'indiques le site plus "cool" où je pourrai acheter (en-ligne) des CD de musique.
>
> *Agent A:* Je peux te donner l'adresse de ce site pourvu que tu me dises ce que tu as acheté.
>
> *Agent B:* D'accord.

Le nombre de participants et les interactions possibles peuvent aussi varier:

- négociation un-à-un: un agent négocie avec un autre, par exemple dans le cas ou on essaie de négocier le prix d'achat d'une maison avec le représentant d'une agence immobilière
- négociation un-à-plusieurs: un seul agent négocie avec plusieurs autres agents, par exemple les enchères où un agent veut vendre un objet
- négociation plusieurs-à-plusieurs: plusieurs agents négocient avec plusieurs d'autres agents en même temps, par exemple les participants à des enchères électroniques comme celles organisées par eBay (http://www.ebay.com).

4 – Mise en oeuvre

1. Les différents langages utilisés pour la programmation des agents

L'objectif de ce chapitre est de présenter les différents langages de programmation utilisés dans la conception des Agents. Le but n'est pas de faire un cours complet sur les langages mais de permettre de pouvoir comprendre la philosophie de la programmation orientée Agent.

1.1. Un mot sur les langages orientés « objet »

La programmation orientée objet (en abrégé P.O.O) a pour objectif de faciliter l'activité de programmation, notamment en permettant de développer des composants logiciel réutilisable.

Elle s'appuie sur l'objet, l'objet est un ensemble regroupant des traitements (méthodes), et des données (propriétés). L'objet permet aux développeurs de penser le problème, non plus en terme uniquement de manipulations de « nombres », mais en terme « d'énoncé» du problème. Dans un L.O.O, le « plan » d'un objet est représenté sous forme d'une classe (contenant des méthodes et des propriétés), plusieurs objets peuvent être créés (instanciés) en s'appuyant sur la même classe. De plus la L.O.O fait appel à des notions tel que l'encapsulation, l'héritage, le polymorphisme.... , qui sont inconnues de la plupart des langages traditionnel tels que le C ou Pascal.

Tout ceci confère aux L.O.O un bon avantage dans les langages de création d'Agent.

1.2. Langage Java

Le langage Java a été créé par la société Sun Microsystems. Le principal objectif de Java est d'augmenter la productivité des programmeurs.

- Le « Write once, run everywhere » : « écrire une fois, utiliser partout » ; pour ne pas être limité aux développements pour une seul plate forme (un programme écrit en java tournera de la même façon sur MS Windows ou sur Unix).
- De plus, il y a de nombreuses fonctionnalités (méthodes) liées aux mondes des réseaux.
- Il est orienté objet, pour faciliter la réutilisation et la maintenance des programmes.
- Pour finir, le minimum nécessaire pour développer une application en langage Java est disponible gratuitement sur le site de Sun Microsystems.

Mécanisme général du langage Java

Pour qu'un programme écrit en langage java puisse être exécuté sur plusieurs plates-formes, il est dans un 1er temps compilé en bytecode. Le bytecode est un code pouvant être interprété par une machine virtuel, la JVM (Java Virtual Machine). Pour chaque plate-forme, une JVM a déjà été écrite par la société Sun Microsystems.

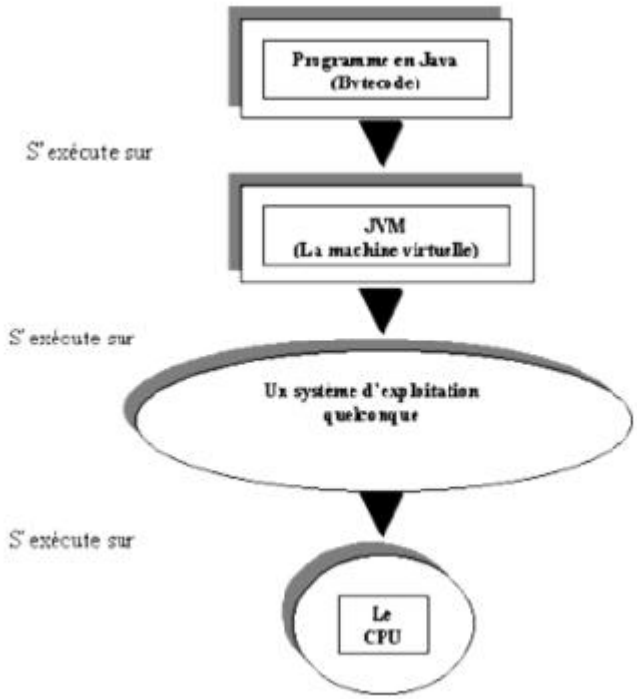

Figure 12 : le mécanisme du langage java

Comme on peut s'en douter, l'interprétation du bytecode par la machine virtuel (JVM) entraîne un ralentissement du programme. C'est pour cela qu'il est possible pour l'utilisateur du programme java d'utiliser un compilateur JIT (Just In Time). Celui-ci permettra à la machine virtuelle de compiler le programme bytecode en code natif du CPU (langage machine) avant son exécution. Le programme ne sera plus interprété par la JVM mais par le processeur de la machine grâce à une compilation qui aura eu lieu « juste à temps » (JIT).

Les packages Java

Pour les développeurs, Java offre la possibilité de regrouper les objets créés (plus exactement les classes des objets) dans un même fichier. Ce dernier est appelé un package, ceci permet de réutiliser au maximum le code déjà écrit.

Critique du langage Java

Une des plus importantes critiques qu'on pourrait faire à Java est son manque de compatibilité entre les versions, malgré les efforts de Sun Microsystems pour l'assurer.

Java et les Agents

Comme nous l'avons vu dans la section à propos des mécanismes, un Agent mobile à besoin d'une machine virtuel pour fonctionner or Java en possède une. De plus le langage Java est sécurisé car c'est la JVM qui vérifie au moment de l'exécution, la fiabilité du code.
Pour finir Java est orienté objet et tout ceci confère à Java une bonne place pour la construction d'Agents.

1.3. Le langage C

Ce langage a été créé en 1972 par Denis Ritchie dans le but d'écrire le système d'exploitation (UNIX). Son succès international a amené l'ANSI (American National Standard Institute) à définir un « standard » (on le qualifie souvent de « C ANSI »).

Le langage C présente les caractéristiques suivantes :

- Le suivi des règles de la programmation structurée
- Un langage orienté système, c'est à dire qu'il est le langage le plus proche du « raisonnement » d'un ordinateur, après l'assembleur
- Le langage C est un langage compilé, c'est à dire que la totalité du code source est traduit en langage machine, avant exécution
- Pour finir le langage C est connu d'un grand nombre de développeurs.

Critique du langage C

Un gros programme écrit en C est difficile à maintenir. De plus, ne s'exécutant pas sur une machine virtuel, celui-ci doit être recompilé pour chaque plate-forme. Par contre, les grands avantages du C sont, qu'il existe quasiment un compilateur C pour chaque type de plate-forme et qu'il est toujours très employé.

Le langage C et les Agents :

Le langage C étant un langage système tout lui est possible. Ce qui fournit aux Agents écrits en C, une très grande puissance et une vitesse d'exécution inégalée. Par contre, cet avantage entraîne deux problèmes :

- Le langage C n'est pas sécurisé, ce qui peut entraîner, un « plantage » des machines qui exécuteraient un processus Agent basé sur ce langage,
- Le langage C devant être compilé avant son exécution, celui ci permet difficilement à première vue de créer un Agent mobile. Ce problème est résolu par certains outils de création d'Agent, par la diffusion du code source du programme Agent et d'un script de « commande » (pas uniquement un makefile). Si la plate-forme ne correspond pas à l'original, alors le programme C est recompilé pour créer un Agent pouvant fonctionner sur cette nouvelle plate forme.

1.4 Le langage C++

Ce langage a été créé à partir des années 1980, par B. Stroustrup (AT&T). Il avait comme objectifs d'ajouter au langage C des spécifications supplémentaires lui permettant d'appliquer les concepts de la P.O.O (programmation orienté objet).

Le langage C++ présente les caractéristiques suivantes :

- Langage compilé,
- Langage orienté objet (sauf pour les puristes de l'objet).

Critique du langage C++

Malgré ces caractéristiques objet, un programme écrit en C++ est difficilement maintenable, ressemblant très rapidement à « une usine à gaz ». Par contre un grand nombre des avantages du langage C s'y retrouvent, avec en plus, la puissance d'un L.O.O.

Les Agents et le langage C++

La puissance du C++ est le plus souvent utilisé pour créer des Agents avec un haut degré d'intelligence voir le site de UMPRS où les sources d'un outil Agent orienté Intelligence Artificielle écrit en C++ sont téléchargeables.

2. Programmation orientée agents

L'architecture de l'agent une fois décidée, on peut utiliser n'importe quel langage de programmation pour implémenter l'agent. Il serait toutefois souhaitable d'avoir un langage de programmation qui offre des constructions spéciales pour bâtir nos agents et qui permette, de cette manière, de développer plus vite nos applications. Les langages de programmation agents ont été conçus dans ce but. Pourtant il est bien difficile de concevoir un langage qui offre des constructions adéquates et efficaces pour implémenter n'importe quel agent. On a vu dans les chapitres précédents qu'il y a une grande diversité d'architectures possibles pour les agents et que les concepts mêmes sur lesquels on conçoit nos agents peuvent être très différents. Les chercheurs ont proposé de nombreux langages pour la programmation des agents, chaque langage favorisant une vision particulière de la notion d'agent intelligent. On ne peut pas dire qu'il y ait actuellement un tel langage généralement accepté. Une place particulière dans le panorama est occupée par le langage de programmation qui a introduit l'approche de la programmation orientée agents. C'est ce principe de langage que nous allons maintenant développer.

La **programmation orientée agents** a été proposée par Yoav Shoham en 1993 comme un nouveau paradigme de programmation, que l'on peut voir comme une spécialisation de la programmation orientée objets. Dans cette approche, les agents sont les éléments centraux du langage, de la même façon que les objets sont centraux pour les langages orientés objets. La perspective sur les agents est cognitive : les agents sont caractérisés par des notions mentales comme leurs croyances, leurs décisions et leurs obligations. De plus, à chaque agent est associé un ensemble d'habiletés qui représentent ce que l'agent sait faire. En même temps, la programmation orientée agents suppose qu'on va développer des programmes dans lesquels plusieurs agents interagissent, ce qui met l'accent sur la dimension sociale des agents. Le langage de programmation proposé par Shoham comme démonstration de ce nouveau paradigme s'appelle AGENT0.

La **différence principale** entre un tel langage et un langage de programmation classique que l'on pourrait utiliser pour développer des agents, vient du fait que les notions mentales qui caractérisent les agents apparaissent dans le langage lui-même, et que la sémantique du langage est intimement liée à la sémantique de ces notions mentales. La programmation orientée agents peut être vue comme une spécialisation de celle orientée objets parce que les modules du programme sont maintenant des agents, c'est-à-dire des objets avec un état qui définit les notions mentales associées, et que les messages entre objets sont remplacés par des messages entre agents. En quoi les messages entre agents différent-ils de ceux entre objets ? Premièrement, parce que ces messages sont modélisés en partant de la théorie des actes de parole, qui s'intéresse aux actions de communication comme informer, demander, offrir, accepter, rejeter et d'autres (pour la théorie des actes de paroles, souvenez vous ce qu'on a présenté dans le chapitre 1 ou lisez le chapitre 4). Deuxièmement, puisque les agents sont autonomes et dotés de capacités mentales, ils ont la liberté de décider s'ils vont ou non exécuter l'action spécifiée dans le message. Par contraste, un objet recevant un message va toujours exécuter l'action spécifiée dans le message. Cette spécialisation même fait que la programmation orientée agents est différente de celle orientée objets comme le montre le tableau ci-dessous, tableau de différences établi par Shoham.

Programmation Orientée Objets / Programmation Orientée Agents

	POO	POA
Unité de base	**objet**	**agent**
Paramètres définissant l'état de l'unité de base	**pas de contraintes**	**croyances, décisions, obligations, habiletés**
Processus de calcul	**envoi de messages et méthodes pour la réponse**	**envoi de messages et méthodes pour la réponse**
Types de messages	**pas de contraintes**	**informer, demander, offrir, promettre, accepter, rejeter, …**
Contraintes sur les méthodes	**pas de contraintes**	**consistance, vérité, …**

Avec l'évolution des théories concernant les agents et des langages de programmation associés, on peut également souligner d'autres différences entre POO et POA :

- les agents sont autonomes alors que les objets ne le sont pas ; un agent va décider par son propre processus de décision s'il exécute ou non une action requise ;
- les agents ont leurs propres buts et ils agissent d'une manière pro-active pour atteindre leurs buts (par exemple, ils saisissent des opportunités) alors que les objets ne le font pas ;
- les agents sont capables d'un comportement social : ils peuvent s'engager dans des interactions complexes, par exemple coopération, compétition, négociation, avec d'autres agents ; ce n'est pas le cas des objets ;
- un système multi-agents est, normalement, un système dans lequel les agents correspondent à des chemins d'exécution séparés ; chaque agent a son propre chemin d'exécution alors que les objets, à part les objets concurrents, ne présentent pas cette caractéristique.

Le débat sur la différence entre agents et objets est encore ouvert, et on peut ajouter d'autres arguments à ceux mentionnés ci-dessus. En parcourant ce cours, vous pourrez décider par vous-mêmes quelles sont les différences les plus importantes, ou même s'il y a des différences.

5 – Perspectives d'avenir

1. Les Agents perceront-ils ?

Du fait des problèmes de mise en place des Agents (aux niveaux recherche, technologie et financement), il n'y a aucune application importante basée sur les Agents. Ces difficultés concernent 4 points : intelligence, mobilité et sécurité, conception de méthodologie et d'outils, avoir des Agents génériques. Pour ces raisons, la technologie Agent se trouve dans une phase critique, malgré des perspectives immenses.
L'évolution rapide des technologies réseaux et des ordinateurs, ainsi que la croissance des informations disponibles sur Internet, amènent à penser que des centaines de millions de personnes seront connectées, à travers leur ordinateur au travail, à l'école, à la maison mais aussi à travers leur télévision, leur téléphone ou leur voiture, de partout.
Pour permettre cela, les Agents mobiles seront un outil essentiel. Les Agents mobiles sont des programmes qui peuvent se déplacer d'une machine hôte à une autre. L'état d'exécution du programme est sauvegardé, transporté sur la nouvelle machine et restauré, permettant au programme de continuer sa tâche. Les Agents mobiles différent des applets et des processus migrants par le fait qu'ils choisissent eux-mêmes la machine « cible » et le moment de leur départ.
Un autre phénomène, qui encourage l'utilisation des Agents mobiles est la limitation du système client/serveur. En effet, ce modèle a l'avantage de permettre le déplacement du client sur des machines plus petites, éloignées et il fonctionne bien pour certaines applications. Cependant il s'écroule dans d'autres situations, incluant les systèmes fortement distribués, les connexions réseau lentes et de mauvaise qualité.

2. Quels sont les obstacles techniques

2.1. Performance et dimensionnement

Les systèmes d'Agents mobiles permettent d'épargner la bande passante au détriment de la charge sur la machine hôte. Car les Agents sont écrits dans un langage lent et interprétable, pour des raisons de portabilité et de sécurité et car ils doivent continuer leur exécution dès leur arrivée. Ainsi, sans déconnexion, les Agents mobiles mettent plus de temps à exécuter que des applications classiques.
Heureusement, des progrès significatifs ont été fait dans la compilation juste-à-temps (pour Java) et dans d'autres techniques, qui permettent au code mobile de s'exécuter comme une application native. De même, des groupes de chercheurs explorent de nouveaux moyens pour réduire le coût de la migration.
Ces effets devraient amener à ce que les Agents mobiles ne soient pas plus gourmands que des applications classiques.
Par exemple : Le système Mozart utilise la programmation avec le langage OZ. Les temps de traitement par rapport à une application Java équivalente ont été réduits.

2.2. Portabilité et standardisation

Presque tous les systèmes d'Agents mobiles permettent à un programme de se déplacer librement parmi des machines hétérogènes. Le code est compilé dans une représentation indépendante de la plate-forme puis, à son arrivée, compilé en code natif ou exécuté dans un interpréteur. Pour avoir des Agents largement répandus, le code devra être portable. Il faudra faire un effort important de standardisation pour obtenir cela. Une première approche existe avec le standard MASIF de l'OMG, mais ne concerne que les communications inter-systèmes et l'administration. La communauté des Agents mobiles devra poursuivre ce début de standardisation, en choisissant une certaine machine virtuelle, ainsi que des formats de code.

2.3. Sécurité

C'est possible de déployer un système d'Agents mobiles qui protège la machine contre les Agents malfaisants. Mais quelques défis restent à relever tels que : protéger les machines sans limiter les droits d'accès des Agents, protéger les Agents des machines malfaisantes et protéger les groupes de machines. Plusieurs groupes travaillent sur ces 3 problèmes, qui devront être résolus pour toutes les applications sur Internet.

3. Quelques mots sur la sécurité et l'éthique des agents.

Dans une société basée de plus en plus sur l'information pour prendre des décisions quelque fois vitales, les agents constituent une réelle menace pour un environnement informatique. En effet les Agents intelligent étant des processus autonomes et pouvant se déplacer sur le réseau, ils peuvent générer des failles de sécurité (destruction d'informations, espionnage, contournement des protections etc.) dont les causes sont les suivantes :

La délégation : On délègue à un Agent une partie de l'autorité de son utilisateur, mais le problème est que l'Agent n'a pas « d'états d'âme ».

La confiance : Une des motivations principales de l'utilisation des Agents et d'éviter à l'utilisateur de faire un travail répétitif, mais un humain fait attention aux effet de bord et aux résultats qu'il obtient, contrairement à un Agent.

La mobilité : Les Agents se déplacent sur les réseaux d'une façon autonome. Comment suivre l'action d'un grand nombre d'agents sur le réseau ?

Les vers : « Les vers sont des programmes autonomes capables de se répliquer en réseau et de se lancer à distance ». Or cette définition ressemble beaucoup à celle des Agents mobiles. Comment reconnaître un vers d'un Agent ?

Pour contourner ces problèmes, les Agents doivent être créés par des outils sécurisés. C'est à dire, un langage dont toutes les possibilités pouvant créer un problème dangereux sont éliminés. Mais cette solution entraîne forcément une perte de la puissance de l'agent.

De plus, l'Agent pose des problèmes éthiques. Imaginez une personne utilisant son Agent pour espionner ou pire, répandre (sous forme de propagande) des idéologies plus que douteuses (les thèses révisionnistes par exemple). Ce dernier type de problème sera difficile à résoudre et pose la même question que la génétique actuellement : Jusque où doit-on aller dans l'application du savoir ?

6 – Conclusion

De notre travail, nous pouvons tirer quelques enseignements :

- La technologie Agent est porteuse de beaucoup d'espérances.
- Celle-ci est en pleine ébullition. Beaucoup d'annonces et de conférences ont lieu sur les principaux sujets.
- De nombreux acteurs sont impliqués dans cette technologie. Pourtant, malgré les efforts déployés, il n'y a pas encore eu l'émergence d'un ou de plusieurs chefs de files. Parmi les acteurs, il y a les grands de l'informatique (IBM) et des télécommunications (British Telecom), mais aussi de petites sociétés. On retrouve aussi des acteurs du monde GNU.
- À part les agents de recherche, les agents d'information et les agents pour le commerce électronique, il ne semble pas y avoir de produits matures. A cause des problèmes de sécurité, les agents mobiles sont encore restreints dans leurs déplacements par la disponibilité des serveurs d'accueil et ne sortent pas trop des laboratoires.
- Du fait de la jeunesse du sujet, la documentation est très dispersée parmi plusieurs sites et quelques livres. Un site en amenant un autre, une question reste posée : Est-ce que je n'ai pas oublié quelque chose ?
- A cause de cette jeunesse, beaucoup de mots importants sont mal définis. Il a fallu passer du temps à qualifier certains termes.

Pour terminer, citons Mr Orfali dans son « guide de survie du client/serveur » :
« Des Agents itinérants. Le nouvel environnement sera peuplé d'Agents électroniques de toutes sortes. Les consommateurs disposeront de leurs propres Agents personnels chargés de veiller sur leurs intérêts ; les entreprises déploieront des Agents pour vendre leurs produits sur le réseau ; des Agents « fouineurs » seront en permanence sur le réseau, collectant des informations pour assurer l'administration des systèmes ou plus simplement, pour établir des statistiques et des tendances. La technologie des Agents comprendra des moteurs de création de scripts multi plate-formes, le workflow et des environnements à code mobile, de type Java, pour permettre aux Agents de vivre sur toute machine du réseau. »

Webographie

http://www.anacip.com/	Modèle d'analyse sémantique et relationnelle
http://www.agentland.fr/	Le portail des agents intelligents : actualité, dossiers et téléchargement
http://www.sam-mag.com/archives/humain.htm	Article : Intelligents mais pas encore humains
http://turing.cs.pub.ro/auf2/	Cours sur les agents intelligents
http://www.damas.ift.ulaval.ca/~coursMAS/	Cours sur les agents et les systèmes multi-agents
http://www.decisionnel.net/agentintelligent/	Agents intelligents
http://www.enpc.fr/enseignements/Legait/projet/victor/chercher/Agent.Int.html	Recherche d'informations et veille sur Internet
http://big.chez.tiscali.fr/gbonnet/laii.html	Les agents intelligents sur Internet
http://mycgiserver.com/~huault/Aglets.pdf	Les aglets
http://www.dess-taii.univ-avignon.fr/pages/PagesEnseignements/Cours/AgentsIntelligents.pdf	Les agents intelligents en Java
http://c.asselin.free.fr/french/agentintelli.htm	Typologie des outils de veille
http://www.geoscopie.com/sources/internet/g051motint.html	Moteurs spécialisés sur Internet
http://dmoz.org/World/Fran%C3%A7ais/Informatique/Intelligence_artificielle/Agents/	Liste de site spécialisés
http://www.limsi.fr/Individu/jps/enseignement/examsma/examsma.htm	Tutoriaux thématiques
http://www.irit.fr/ACTIVITES/EQ_SMI/SMAC/	Systèmes multi-agents coopératifs
http://agents.umbc.edu/	Information sur la technologie agent
http://www.agentintelligent.com/	Veille technologique, chatterbot et agents intelligents